PROJECTE DE FI DE CARRERA

TÍTOL: Disseny i implementació de software per la xarxa audiovisual IP de Catalunya (XAC)

AUTOR: Sílvia Pinel Jiménez

DIRECTOR: Jesús Alcober Segura

DATA: 11 de juliol de 2006

Títol: Disseny i implementació de software per la xarxa audiovisual IP de Catalunya (XAC)

Autor: Sílvia Pinel Jiménez

Director: Jesús Alcober Segura

Data: 11 de juliol de 2006

Resum

Aquest projecte es troba dintre del marc d'un projecte que porten a terme diferents grups d'empreses i universitats, anomenat XAC (Xarxa audiovisual de Catalunya). L'objectiu del projecte XAC és la compravenda de continguts audiovisuals utilitzant la xarxa d'Internet distribuïda: P2P. Dintre d'aquest context, en aquest projecte es dissenyaran e implementaran tres serveis que s'utilitzaran a l'aplicació final: Upload/Download i Streaming d'un contingut d'un servidor en concret o sobre la xarxa P2P.

Title: Software design and implementation for IP audiovisual network of Catalonia (XAC)

Author: Silvia Pinel Jiménez

Director: Jesús Alcober Segura

Date: July, 11th 2006

Overview

This project is done between different companies and universities and it is calling XAC (Xarxa Audiovisual de Catalunya). The aim of the XAC project is buying and selling audiovisual contents using the distributed Internet network: P2P. Inside this context, concretely in this project three WEB Services that will be use for the final application will be designed and implemented: Upload/Download and the Streaming of content from a specific server or over a P2P network.

A David, a Pedro Díaz, a Pedro Lorente,
al grup de la UPF, al Toni Oller, i en especial,
a Jesús Alcober i a Alberto José Gónzalez
(per la seva paciència).

ÍNDEX

Índex d'il·lustracions

Introducció

El projecte XAC consisteix en una plataforma pilot encarada al sector audiovisual de Catalunya, i amb l'objectiu de facilitar l'intercanvi (compravenda) de continguts. Aquest projecte va ser proposat per diferents socis de la Fundació i2CAT per desenvolupar-lo al llarg de l'any 2005 i part del 2006.

El paquet de treball en el qual s'emmarca la proposta del projecte inclou la definició i implementació d'un prototipus que permeti l'intercanvi segur de continguts audiovisuals entre diversos agents, en un entorn digital.

El prototipus resultant d'aquest paquet de treball haurà de permetre la compravenda segura de continguts audiovisuals entre empreses proveïdores de continguts, tenint en compte els drets associats a aquests continguts, mitjançant un sistema de gestió de drets digitals (DRM).

Per fer més atractiu l'intercanvi estandarditzat de continguts audiovisuals, s'afegiran uns quants serveis addicionals independents que incrementaran la qualitat del servei global. Això inclou la facilitat de transcodificació de continguts fent servir tecnologies Grid, presentació dels continguts, i diverses tècniques per millorar l'eficiència dels protocols de transferència.

Per tal de dur a terme l'intercanvi de continguts entre entitats, un cop superada la part de negociació de drets, cal un sistema eficient que els faci arribar als receptors. Partint del supòsit que cada entitat tindrà emmagatzemats en local aquests continguts, es poden usar dues filosofies ben diferenciades:

En primer lloc es pot utilitzar un sistema de transmissió tipus client-servidor tradicional, en que es realitza una connexió contra un servidor i es descarrega el contingut i aquí finalitza tota la interacció. Aquest sistema seria basat en el protocol FTP (File Transfer Protocol) tradicional, amb una fiabilitat i robustesa àmpliament demostrades al llarg dels anys que fa que funciona. L'inconvenient que té aquest sistema és la càrrega que suposa per al servidor la descàrrega contínua de fitxers.

Per tal de solucionar-ho es poden usar altres sistemes anomenats peer-to-peer. Aquests sistemes presenten l'avantatge que un cop feta la descàrrega, el contingut està disponible des de dues fonts diferents, de manera que s'optimitza el temps de descàrrega. Si el contingut es troba a diversos servidors, lògicament, el temps necessari per a obtenir el contingut disminueix, alhora que disminueix la càrrega dels diferents servidors. Al disminuir la càrrega de treball dels servidors de continguts aquests poden, alhora, oferir un major nombre de connexions, i per tant absorbir un major nombre d'usuaris. Un altre avantatge que presenten aquests sistemes són la alta disponibilitat dels continguts, ja que si pel motiu que sigui cau un dels servidors, el contingut segueix estant disponible a d'altres localitzacions, de manera que no hi ha necessitat d'espera a que se solucioni el problema en aquell extrem i permet un treball més àgil.

Basant-se en aquesta segona concepció es poden usar protocols peer-to-peer com el conegut BitTorrent, amb una alta implantació per tota la xarxa d'Internet, o el protocol JXTA utilitzat per la Fundació i2CAT, dins del marc del projecte integrat.

Degut a que el protocol JXTA s'ha fet servir al Projecte Integrat per a implementar l'aplicació e-Ruc, la seva implantació dins de la aplicació sembla la més adequada per a les nostres necessitats, oferint una eina global que abastarà tot el procés d'intercanvi des de l'inici fins el final i que redundarà en una major facilitat d'ús de la mateixa als seus consumidors.

L'objectiu d'aquest projecte és dissenyar e implementar els protocols de transferència per a l'aplicació final del projecte XAC, en tres àmbits: servei de Download, servei d'Upload i servei Streaming dels continguts.

Primer s'estudiaran per cada cas, els avantatges i inconvenients que ens comporta l'ús de cada una de les opcions que existeixen per fer les transferències de fitxers, per, finalment dissenyar e implementar l'opció que millor s'adapta a cada servei.

1 CAPÍTOL 1. Projecte XAC

1.1 La Fundació I2CAT i el Projecte Integrat

La Fundació i2CAT (veure [1]) és una plataforma col·laborativa per a la recerca i la innovació. Els objectius principals d'I2CAT són:

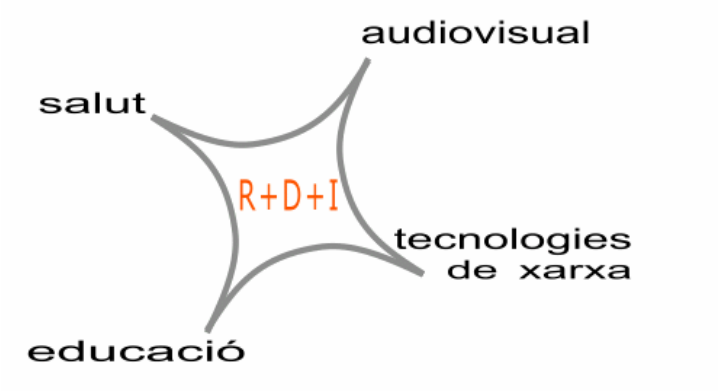
- Desenvolupar projectes de recerca i innovació.
- Promoure xarxes de recerca avançades i aplicacions de gran ample de banda, i augmentar-ne l'ús.
- Crear noves plataformes col·laboratives entre el sector privat i la recerca universitària
- Promoure equips de treball entre institucions de la resta del món amb objectius alineats amb els de la fundació dins l'àmbit del 'networking'
- I2cat és un fòrum per aquelles institucions, societats i companyies que desitgin participar en projectes globals pel desenvolupament d'infraestructures I serveis de xarxa de banda ample.

I2CAT ha desenvolupat projectes de recerca que han creat una plataforma experimental de xarxa, serveis de Middleware i aplicacions genèriques de les més avançades d'Europa, i un conjunt de projectes específics per sectors, que anomenem Clústers.

Els Clústers són estructures de recerca estratègica entre universitats, administracions i empreses amb un mateix interès temàtic o sectorial dins el camp de la recerca en Internet avançada.

Els Clústers han permès augmentar la participació de nous socis col·laboradors. Els Clústers temàtics són una agrupació de projectes de serveis i aplicacions digitals sinèrgiques, coordinats i impulsats per les empreses i organismes que han signat un conveni de soci col·laborador amb i2CAT. Inicialment existeixen quatre Clústers, ampliables a altres temàtiques: Tecnologies de xarxa, Audiovisual,Salut i Ensenyament.

Cada Clúster disposa dels serveis oferts per la plataforma i2CAT i paral·lelament, obtindrà internament recursos suficients d'infraestructures, serveis i aplicacions per dur a terme el desenvolupament dels seus propis projectes.



Il·lustració 1. Clúster d'i2CAT

Durant el període 2004-2005 es va dur a terme el disseny i implementació d'un Projecte Integrat (PI) dedicat a cobrir els serveis multimèdia utilitzant tecnologies basades en WEB Services.

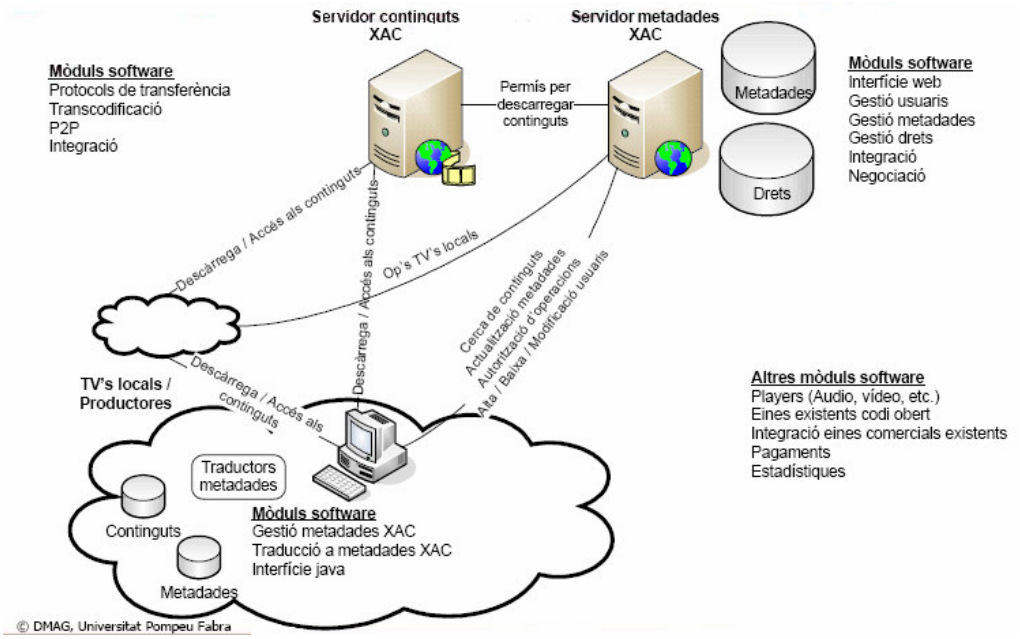
El Projecte Integrat pretén crear un sistema a través del qual un usuari pot accedir des del seu terminal (fix o mòbil) a contingut audiovisual d'alta qualitat a partir de la integració de diferents mòduls que interaccionen entre si de manera transparent per a l'usuari mitjançant serveis WEB. El projecte configura una xarxa d'entorns flexibles per a la creació, edició i distribució de continguts audiovisuals sobre IP per a tothom.

Per a assolir l'objectiu plantejat, el projecte centra els seus esforços en la integració de serveis, localització dinàmica de recursos, heterogeneïtat de serveis, transparència de la xarxa, emmagatzematge, transport i processat de continguts multimèdia, transcodificació de continguts, topologia dinàmica de la xarxa, xarxes fetes a mida, gestió distribuïda dels recursos en xarxa i qualitat de servei. En definitiva, defineix models de xarxa i nous models de negoci.

1.2 Projecte XAC

El projecte XAC (Xarxa Audiovisual de Catalunya) neix de la proposta de desenvolupament d'una plataforma pilot en la xarxa Internet pel sector audiovisual de Catalunya, proposat per diferents socis de la Fundació i2cat. L'objectiu del projecte és la creació d'una plataforma en xarxa IP d'intercanvi de continguts audiovisuals dins la Internet de segona generació.

El projecte utilitzarà la Internet avançada per la producció, distribució i emissió de canals de TV i oferirà a un conjunt de TV locals actuals la possibilitat d'ús de les noves xarxes IP. Pretén facilitar la digitalització de les empreses i televisions de Catalunya, accelerar la seva posada en xarxa, preservar i fomentar la generació de continguts audiovisuals en català i donar projecció al sector audiovisual de Catalunya.



Il·lustració 2. Mòduls i arquitectura del projecte XAC

Els principals objectius que és van fixar al projecte XAC van ser el següents:

1. Proveir de connectivitat IP als diferents actors del projecte (inicialment a les TVs locals)

2. Estandardització de formats de metadades.
Es relacionaran les metadades del proveïdors de continguts amb un estàndard comú. Es considerarà també la informació necessària per a la gestió dels drets digitals (Digital Rights Management).

La recerca a desenvolupar per a aquestes funcionalitats serà al voltant de especificar els mecanismes i conjunts de metadades més adients, tenint en compte els requeriments de les televisions a Catalunya i les iniciatives més avançades a Europa.

3. Intercanvi segur de produccions audiovisuals.
És important definir un sistema d'intercanvi basats en criteris de treball en xarxa i en arquitectures Peer to Peer. Es pot veure com un servei simple de compravenda de continguts audiovisuals amb Digital Rights Management.

Aquesta tasca també inclou activitats de recerca e innovació pel que respecta a les arquitectures per organitzar la compravenda de forma segura i eficient, que tenen força relació amb les peculiaritats dels continguts i la comunicació entre els seus proveïdors.

1.2.1 Paquets de treball del projecte XAC

El projecte XAC està dividit en cinc paquets de treball:

1. Coordinació del projecte i dels informes: aquest primer paquet inclou la direcció del projecte, coordinació de les reunions, de les diferents tasques i elaboració dels diferents informes que es requereixin.
2. Provisió de connectivitat als usuaris del projecte: L'objectiu d'aquest segon paquet és portar a terme la provisió de connectivitat IP de banda ampla, per dotar d'accés als diferents serveis de la plataforma a les seus dels diferents usuaris del projecte acordats per conveni.
3. Estandardització dels formats de dades i metadades: Aquest paquet de treball inclou l'estudi i la implementació de les eines que permetin treballar amb el conjunt de metadades que apliquin als diversos continguts que es podran intercanviar dintre de la plataforma pilot del XAC. També s'inclouran els formats d'intercanvi de dades i el necessari per a la gestió dels drets intel·lectuals.
4. Intercanvi segur del contingut audiovisual: Aquest paquet de treball inclou la definició i implementació d'un prototipus que permeti l'intercanvi segur de continguts audiovisuals entre diversos agents, en un entorn digital.
5. Proves d'avaluació per part de l'usuari, demostracions i promoció: l'últim paquet inclou l'avaluació de les diferents plataformes i serveis de la XAC, demostracions de l'ús de les mateixes i dinamització dels resultats amb la seva promoció.

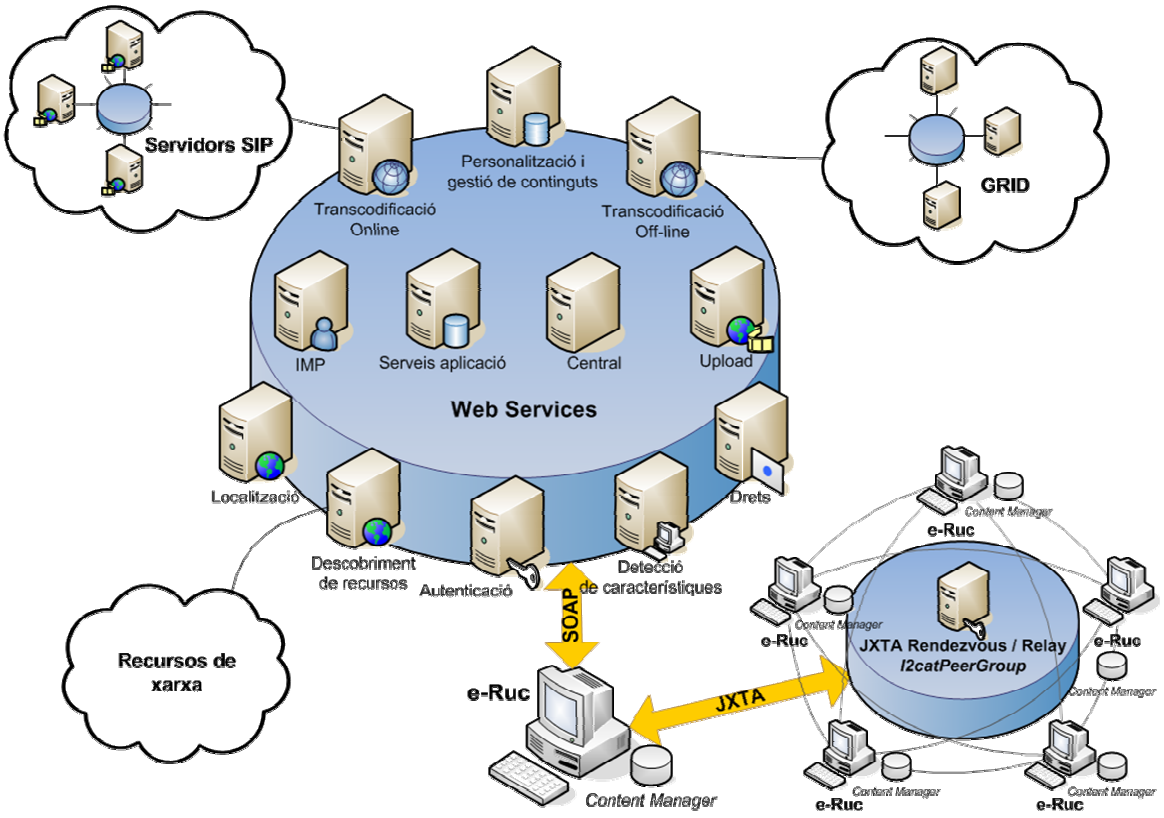
Aquest projecte es centrarà concretament a la part del disseny i la implementació de l'intercanvi de continguts (protocols de transferència de fitxers) que s'engloba dintre del paquet de treball 4, l'intercanvi de continguts. Per fer-ho, s'estudiaran les possibilitats que es poden tenir per dissenyar-ho i finalment s'explicarà pas a pas el mètode seguit per implementar-ho (Capítols 2 i 3).

1.2.2 Aplicació: e-Ruc

Dintre del context del Projecte Integrat es va dissenyar un mòdul anomenat *"e-Ruc"* (veure 8.1 (Annex A)) amb la finalitat de proporcionar una interfície fàcil i amigable per oferir als usuaris una aplicació avançada per compartir i gestionar recursos audiovisuals.

E-Ruc esta format per una arquitectura avançada d'aplicació distribuïda que combina diferents WEB Services pels diferents mòduls amb P2P. Al tractar-se d'una aplicació per a continguts multimèdia, ha estat construït basat a l'estàndard MPEG21[1]. A la Il·lustració 3 es mostra aquesta arquitectura de forma genèrica:

[1] Per més informació sobre l'estàndard: http://www.chiariglione.org/mpeg/index.htm



Il·lustració 3. Arquitectura de l'aplicació e-Ruc

Aquesta arquitectura està formada per una sèrie de mòduls amb les següents funcionalitats:

- Autenticació i seguretat: usuari i password amb autenticació P2P per a tots els clients de l'aplicació.
- Gestió de continguts: es permet crear, modificar o esborrar continguts, afegir descripcions semàntiques per una cerca fàcil...
- Gestió de drets: creació de llicencies de distribució i d'usuari, especificant els drets de distribució o d'ús d'un contingut.
- Control de drets i protecció del continguts: mitjançant els mecanismes de xifrat es pot distribuir continguts a la xarxa P2P que només poden ser visualitzats amb la clau de desxifrat correcte.
- Distribució i accés als vídeos sobre IP: sistemes de distribució de continguts de vídeo sobre IP.
- Transcodificació en directe i en diferit dels continguts audiovisual.
- Cerca dels continguts: localització dels continguts depenen uns criteris demanats.

1.2.3 Protocols de transferència

Com ja s'ha dit a la introducció, aquest projecte es basarà en estudiar la part de transferència de fitxer per finalment implementar l'opció que millor s'adapti als requeriments dels projecte XAC.

Dintre del context d'aquests requisits, es fixen tres filosofies ben diferenciades per implementar els serveis: l'Streaming del contingut audiovisual en temps real, l'Upload o càrrega d'un fitxer en un servidor central o a la xarxa P2P i el Download d'un fitxer (descàrrega del contingut al/del servidor o a/de la xarxa P2P).

A continuació es veuran les diferents opcions que es poden considerar per fer la implementació d'aquests serveis i als capítols 2 i 3 s'estudiaran i es dissenyaran els tres serveis per fer més tard, la implementació.

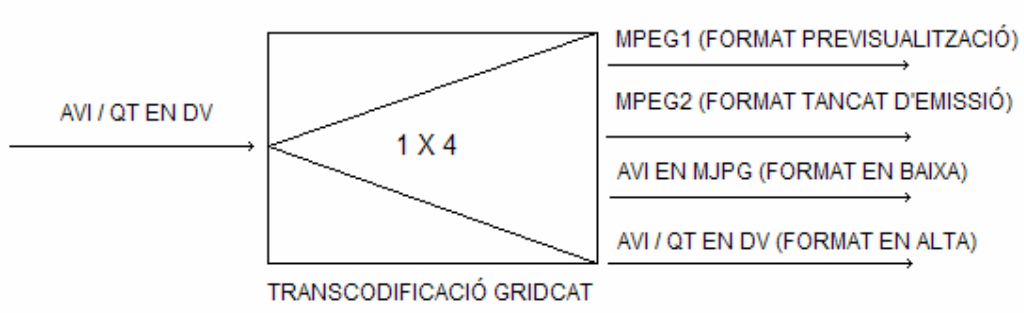
1.2.4 Promoció dels continguts

Com ja s'ha comentat a la introducció d'aquest projecte, l'objectiu final és la de proporcionar un medi fàcil i accessible que faciliti l'intercanvi (compravenda) de continguts a les entitats (en aquest cas, televisions locals) associades al projecte.

Per a que aquest objectiu es dugui a terme, s'ha fet necessari una visita prèvia a cada una de les televisions explicant-li en cada cas l'objectiu que es volia dur a terme, característiques del projecte i el software que s'anava a utilitzar com a aplicació final, l'e-Ruc.

Dintre d'aquesta part, també es va fer un estudi dels formats i còdecs que es farien servir a l'hora de digitalitzar els continguts audiovisuals, per tal d'obtenir uns requisits mínims de qualitat, ubiqüitat i compatibilitat entre totes elles. Per fer-ho, es va fer un compte d'FTP per cadascuna de les televisions i se'ls va demanar que pugessin un vídeo de prova. Amb aquest continguts, es va fer un estudi dels formats i còdecs que s'utilitzaven i es van escollir els que millor anaven per a fer després la transcodificació dels continguts en alta o baixa qualitat, així com el format o còdec de la previsualització del vídeo o l'emissió en Streaming.

A la següent il·lustració es mostra el format escollit finalment, amb el còdec definitiu, que totes les televisions hauran d'utilitzar i que permetrà unificar el sistema de treball, i les transcodificacions que s'utilitzaran per les diferents qualitats dels serveis del projecte.



Il·lustració 4. Formats i còdecs pels vídeos de les televisions

A partir de l'estudi finalment es va establir que el format que s'utilitzaria serà: AVI/QUICKTIME amb códec DV. Amb aquest fitxer QT/AVI DV definitiu és farà la transcodificació vers al dos tipus de MPEG-2 diferents, per previsualització i emissió, i cap al format d'edició en baixa qualitat (amb códec M-JPEG).

Finalment, per als formats de previsualització i d'emissió en la XAC es va optar de forma genèrica en que fossin un MPEG tots dos. Només calia definir el tipus de MPEG que es necessitava en cada una de les dues situacions, que finalment es va establir el MPEG-2.

Pel que fa al format d'edició en baixa qualitat dels vídeos, es va optar pel format M-JPEG, mentre que el format d'edició en alta qualitat només seria una còpia del DV inicial.

1.3 Opcions per la implementació

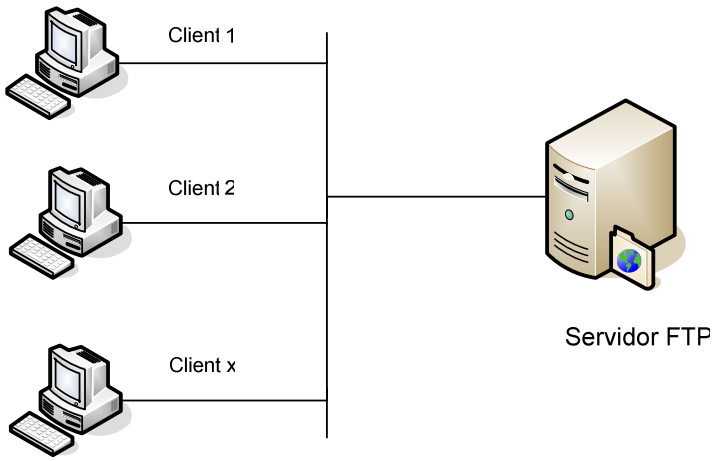
Inicialment es van barallar diferents opcions per fer la implementació d'aquest mòdul, però bàsicament es van estudiar els pros i els contres de fer-ho amb FTP o amb protocols P2P ja que semblaven els mes adients per l'aplicació que es volia dissenyar.

A més a més, es va considerar que el fet de que al PI ja s'havia fet aquest estudi, utilitzar la seva opció ens aportava seguretat (ja que en aquest projecte havia funcionat) i avantatges (ja que es podria aprofitar part de la seva implementació).

Tot i això, s'ha fet un petit estudi de les dues opcions abans esmentades, per decantar-nos finalment per l'opció del PI.

1.3.1 FTP

Al començar a estudiar aquest protocol de transferència de fitxers, el primer que s'ha de tenir en compte és l'arquitectura que utilitza: client-servidor tradicional.



Il·lustració 5. Arquitectura Clients – Servidor FTP

Amb aquesta filosofia i després de tots els estudis que ja estan fets sobre aquesta arquitectura, ja es pot descartar la seva utilització per aquesta aplicació ja que el fet d'utilitzar un únic servidor per a l'emmagatzematge de tots els vídeos i el servei dels mateixos ja aporta l'idea de problemes de congestió en les transmissions i arquitectura poc escalable.

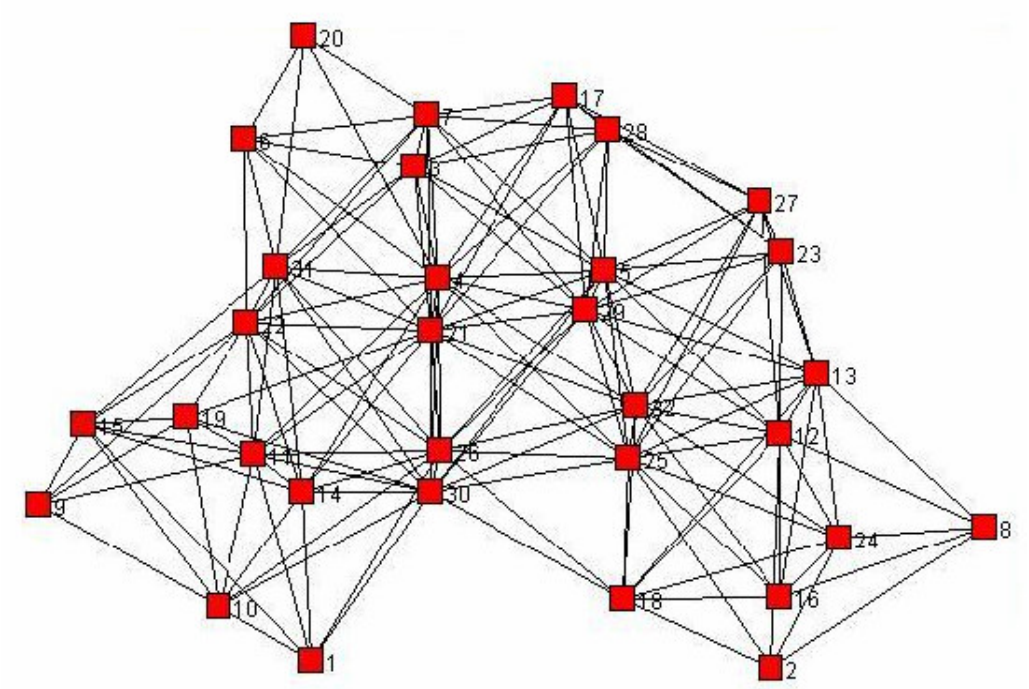
1.3.2 Protocols P2P

L'altre alternativa que hi ha és la utilització de les xarxes peer-to-peer (P2P, veure [2][3]) on és troba una xarxa on no hi ha clients ni servidors, sinó una sèrie de nodes que es comporten a la vegada com clients i servidors dels altres nodes. Aquest model de xarxa contrasta amb el model client-servidor, ja que qualsevol node pot iniciar o completar una transferència compatible.

A l'arquitectura de les xarxes P2P cada estació de treball te capacitats i responsabilitats equivalents, és a dir, son xarxes formades per peers (nodes) on cada node de la xarxa pot actuar tant de client com de servidor, i la comunicació o la col·laboració és directa entre ells.

El model P2P té l'avantatge de ser altament escalable ja que qualsevol node sense importar la seva ubicació ni la seva arquitectura, pot unir-se a la xarxa i actuar amb la resta de peers.

A la següent imatge es pot veure la filosofia que segueix una xarxa P2P, en la que tots es nodes es poden comunicar entre ells:



Il·lustració 6. Esquema xarxa P2P

1.3.2.1 Avantatges e inconvenients

Amb aquesta nova xarxa, un dels principals avantatges que s'aconsegueix és no haver de disposar d'un servidor central per a la interconnexió dels peers o l'emmagatzematge dels continguts.

El fet de prescindir d'aquest servidor central, ens permet crear una xarxa distribuïda on la informació i els continguts es troben als diferents participants, fent-la llavors molt més escalable que la estructura típica amb el servidor centralitzat i, a més, fent un balanceig del tràfic utilitzant la millor xarxa de comunicació. A més, el fet de tindre la informació a diferents elements de la xarxa, fa més accessible l'accés a aquesta informació a més de donar una alta disponibilitat i optimització amb l'ús dels recursos.

Per altra banda, com era de suposar, no és una xarxa 100% perfecta, ja que també te certs inconvenients. Un dels principals és també degut a la descentralització, ja que degut això, fer una gestió de la xarxa és molt més complexa i per això més cara. Per altre costat, si no hi ha una bona gestió, això pot provocar una qualitat de servei més dolenta, que pot donar lloc a pèrdua d'informació, obtenció de dades corruptes, peticions ignorades, col·lisions amb altres serveis...

Tot i tindre aquests inconvenients, dia a dia surten serveis nous amb això ja contemplat que fan que la xarxa P2P sigui el més estable possible i amb una

bona qualitat de servei. És per aquest motiu i continuant amb la filosofia d'utilitzar les aplicacions i serveis més pioners, que s'ha escollit aquesta opció per la transferència de fitxers. Concretament, el protocol utilitzat és JXTA, ja que ha estat l'opció triada al PI i, per tant, considerada com a bona opció també pel projecte XAC. A continuació s'explicarà amb una mica més de detall en que consisteix aquest protocol.

1.3.2.2 Tecnologia JTXA

Els protocols JXTA (veure [4]) comprenen una plataforma de computació en una xarxa oberta dissenyada per a nodes peer-to-peer (P2P). El sistema generalitzat dels protocols JXTA habilita tots els dispositius connectats a la xarxa (incloent telèfons mòbils, PDAs, PCs i servidors) per comunicar-se i col·laborar com a peers. Els protocols JXTA també permeten als desenvolupadors crear i desplegar serveis i aplicacions interoperables a la revolució de les xarxes P2P a Internet.

Els protocols JXTA estandarditzen la forma en que els peers:

- Descobreixen als altres
- S'autoorganitzen en grups de peers
- Anuncien i descobreixen recursos a la xarxa
- Es comuniquen amb els altres

JXTA està dissenyat per ser independent de les implementacions de les capes inferiors. En particular JXTA:

- No requereix l'ús de cap llenguatge de programació ni cap sistema operatiu en especial
- No requereix l'ús de cap xarxa de transport ni topologia en particular
- No requereix l'ús de cap model d'autenticació, seguretat o encriptació

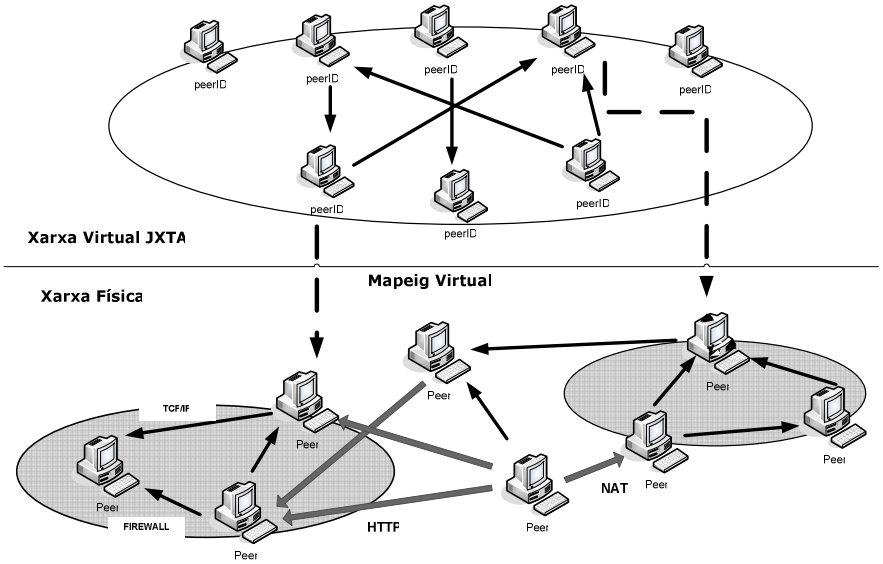
JXTA proporciona una simple i genèrica plataforma P2P amb tots els tipus de funcionaments necessaris per als host i per als serveis de la xarxa:

- JXTA està definit per un determinat numero de protocols. Cada protocol és molt fàcil d'implementar e integrar en serveis i aplicacions P2P. A més, els serveis que ofereix un proveïdor pot ser utilitzat transparentment per una comunitat d'usuaris d'un altre proveïdor.

- Els protocols JXTA estan definits per ser independents dels llenguatges de programació, llavors poden ser implementats en C/C++, Java, Perl, i un gran numero de llenguatges més. Diferents dispositius amb diferents softwares apilats, poden interactuar fent ús dels protocols JXTA.

- Els protocols JXTA estan dissenyats per ser independents dels protocols de transport. Els protocols poden ser implementats sobre TCP/IP, HTTP, Bluetooth, HomePNA [2]i molts altres protocols.

A la il·lustració següent es mostra l'arquitectura que segueix una xarxa que utilitza el protocol JXTA:



Il·lustració 7. Arquitectura de la xarxa virtual JXTA

La xarxa física de la il·lustració mostra l'arquitectura de dues xarxes d'ordinadors connectades a través de dos peers dintre de la xarxa P2P. A partir d'aquesta arquitectura surt la xarxa virtual que es crea amb els protocols JXTA en la que tots els peers estan connectats entre sí i poden localitzar-se i interactuar entre ells.

1.3.2.2.1 Projecte JXTA

El projecte JXTA es troba dividit en tres nivells:

- Plataforma. Aquesta capa encapsula els termes bàsics comuns a les xarxes P2P, incloent peers, grup de peers, descobriment, comunicació, monitorització i seguretat. Aquesta capa es comparteix idealment entre tots els dispositius P2P i fa que la interoperabilitat entre ells sigui possible.

[2] HomePNA permet crear una xarxa local a través de la línea telefònica (http://www.homepna.com/)

- Serveis. Aquesta capa inclou serveis de xarxa que no són absolutament necessaris per a que les xarxes P2P puguin operar però moltes vegades són desitjables a entorns P2P. Alguns exemples de serveis de xarxa inclouen cerca i indexació, directori, sistemes d'emmagatzematge, compartició d'arxius, sistemes de distribució d'arxius, lloguer del recurs, translació del protocol, autenticació i servei de PKI[3]

- Aplicacions. Aquesta capa inclou missatgeria instantània P2P, gestió de continguts d'entreteniment i entrega, sistemes d'e-mail P2P, sistemes de subhastes distribuïdes, i molts altres. Òbviament, el límit entre serveis i aplicacions no és rígid. Una aplicació d'un client pot ser considerada com a servei per un altre client.

1.3.2.2.2 Avantatges e inconvenients de JXTA

Com avantatges més importants de la utilització dels protocols JXTA es pot parlar, com ja s'ha parlat en paràgrafs anteriors, de la interoperabilitat, que consisteix en que totes les aplicacions de l'entorn utilitzen la mateixa plataforma de comunicacions (el model de comunicació de cada aplicació pot ser diferent).

Una altre característica important en la utilització d'aquests protocols és la independència amb els diferents sistemes operatius, el llenguatge de programació o l'entorn de desenvolupament.

Però no tot són avantatges amb el JXTA. Hi han diferents discussions degut a la seva flexibilitat. Tot i que JXTA utilitzi especificacions XML per a tots els aspectes de comunicació i per a totes les aplicacions P2P, JXTA no pot satisfer a una aplicació independent de P2P ja que a una aplicació independent l'*overhead* de la xarxa amb la utilització de la missatgeria XML és mes un problema que no pas una solució, especialment si el desenvolupador no te la intenció d'aprofitar les capacitats de JXTA per incorporar altres serveis P2P.

Altres punts de JXTA que es critiquen és l'abstracció de la xarxa de transport pel fet que ara la majoria d'aplicacions P2P utilitzen a dia d'avui el protocol de transport TCP. Es planteja llavors si es necessari de crear un nou protocol de transport afegint *overhead* a la xarxa en comptes d'utilitzar TCP.

Pero, com ja s'ha comentat, tot i tindre aquest inconvenients, aquest protocol s'ajusta a les necessitats d'aquest projecte i per tant serà l'utilitzat a la implementació.

[3] Public Key Infrastructure (PKI) http://www.pki-page.org/

2 CAPÍTOL 2. Disseny

2.1 Introducció

En aquest capítol es farà la descripció i el disseny dels protocols de transferència de fitxers explicada ja a la introducció. Per fer-ho, es dividirà el capítol en dos apartats: adaptació dels mòduls que es troben al Projecte Integrat (PI) i que serveixen per a la XAC i implementació de les especificacions pròpies de la XAC.

A més a més, en aquest capítol es descriurà el funcionament de cada servei dissenyat, i es veuran les possibilitats per crear una arquitectura estable i amb certa gestió.

2.2 Adaptació dels mòduls del PI a la XAC

Dintre del marc del Projecte Integrat la part que concretament s'adaptarà al projecte XAC és la que s'encarrega d'implementar una arquitectura genèrica per serveis de media basats en protocol SIP, que permet la transcodificació de protocols i continguts. Concretament, dintre d'aquesta part s'adaptaran les funcionalitats que ens permetin implementar els serveis de "*Upload*" d'un fitxer a la xarxa o a un servidor en concret o l'"*Streaming*" d'un recurs en concret.

El mòdul del PI que s'estudiarà a continuació, està dividit en dues parts: transcodificació de protocols i transcodificació de continguts. Dintre de la part de transcodificació de continguts s'utilitzaran les parts que permetin fer l'Streaming i el Upload d'un contingut, i que utilitzarà la part de transcodificació de protocols per oferir un servei combinat de, per exemple, la distribució en temps real.

Les funcionalitats d'aquest mòdul que s'aprofitarà del PI són les següents:

- Transcodificació en temps real per a adaptar-se a les especificacions o limitacions del client.
- Informació completa de transcodificadors i servidors de media.
- Llista de recursos disponibles, per exemple: vídeos d'alta qualitat i les seves variacions (transcodificacions).
- Actualització e inserció de recursos.
- Descàrrega de recursos en temps real.

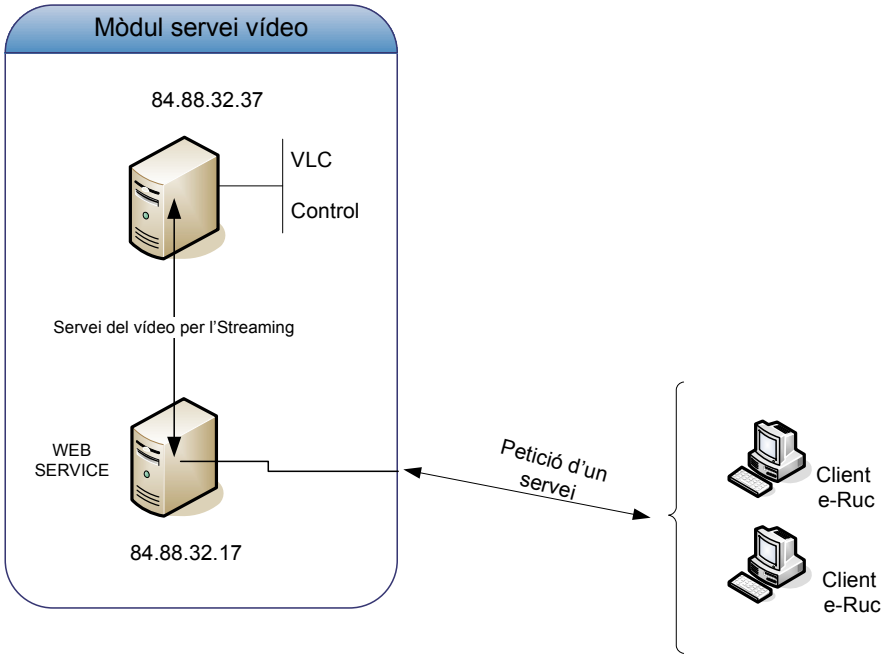
D'aquestes funcionalitats s'eliminaran els serveis que per la XAC no són necessaris i s'adaptaran per acabar tenint els serveis de Upload i Streaming requerits a les especificacions.

Tant tots els mòduls del PI com tot els mòduls del projecte XAC s'han fet utilitzant les següents tecnologies:

- Màquina virtual de Java versió 1.5
- Kit de desenvolupament de serveis WEB: Apache Axis[4]
- Servidor d'aplicacions: Apache Jakarta-Tomcat[5]

2.2.1 Esquema general

L'esquema que actualment utilitza el PI pel que fa als serveis que es volen aprofitar per la XAC, és:

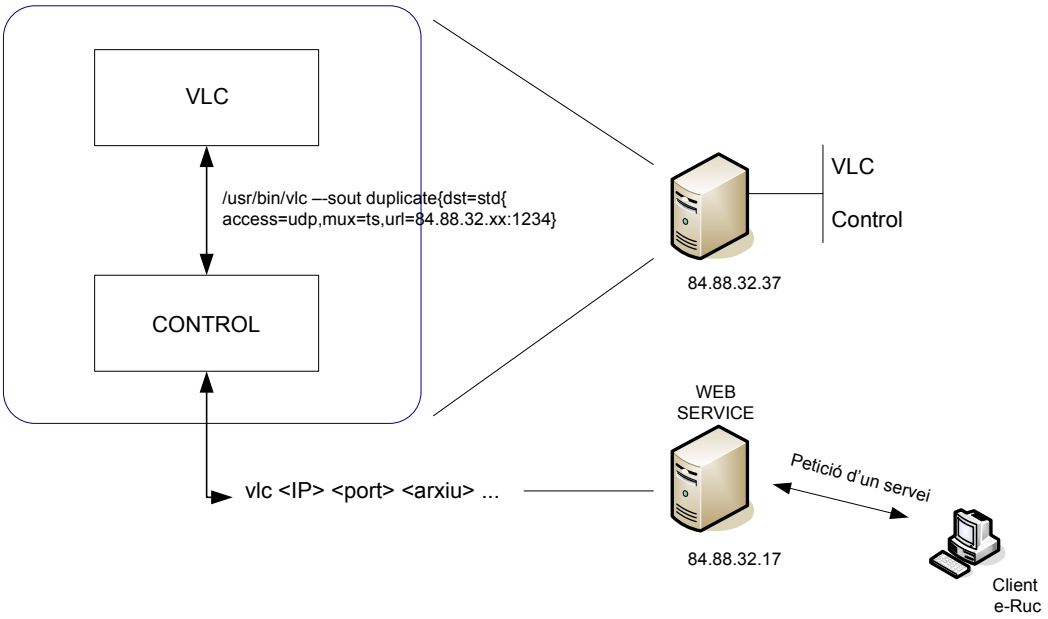


Il·lustració 8. Esquema del muntatge del servei d'Streaming del PI

A la Il·lustració 8 es veu l'esquema general muntat pel PI on els clients fan una petició d'un dels serveis disponibles i és el WEB Service qui s'encarrega de gestionar la petició i executar una o altre comanda depenen del servei demanat.

En el cas del servei d'Streaming, haurà d'interactuar amb el servidor VideoLAN, que conté la capa de Control i el servidor propi VLC, per poder retornar el vídeo. A la Il·lustració 9 es mostra aquesta interacció:

[4] Per més informació: http://ws.apache.org/Axis
[5] Per més informació: http://jakarta.apache.org/



Il·lustració 9. Interacció entre el WEB Service i el Servidor VLC

L'esquema mostra com els clients fan la petició d'un vídeo a partir de l'aplicació e-Ruc. És llavors el WEB Service (84.88.32.17) qui crea la comanda per cridar al VideoLAN i l'envia a un altre equip on està el servidor de VideoLAN (84.88.32.37) el qual té una API de control que fa de traductor per a que ho pugui entendre el servidor VLC. El mateix servidor VideoLAN serà també qui busqui i retorni (en aquest cas) el Streaming del vídeo a la IP que l'ha demanat.

Aquest mateix sistema és l'encarregat també de fer la transcodificació dels recursos, en funció de les especificacions o limitacions dels clients. Aquesta part, però, ocupa un mòdul sencer dintre del projecte XAC, però que no és l'objectiu d'aquest projecte, per tant, serà una part a eliminar.

Pel que fa al servei d'Upload, l'esquema és el mateix que al cas de l'Streaming, però sense la utilització del servidor VideoLAN, ja que únicament el WEB Service haurà de tractar la petició que farà un client de pujar un vídeo i emmagatzemar-ho al lloc corresponen.

L'aplicació l'e-Ruc contindrà els dos serveis descrits abans a més de les funcionalitats descrites a l'apartat 1.2.2. Paral·lelament a la implementació de l'e-Ruc, hi ha un grup dedicat a fer un portal WEB que tindrà les mateixes funcionalitats que l'aplicació, però sense la necessitat de tindre el programa. Aquesta part però la fa un altre grup del projecte XAC i per tant no és l'objectiu d'aquest PFC.

A continuació s'explicarà d'estructura que segueix el mòdul del PI, especificant quines parts es podran aprofitar per la XAC i quines no.

2.2.2 Mòdul de transcodificació del PI

El mòdul de transcodificació és accessible a través d'una interfície pública basada en serveis WEB. Hi ha 2 Serveis WEB implementats dins del mòdul:

- Servei WEB per demanar i controlar continguts de media.
- Servei WEB per a la publicació de continguts.

2.2.2.1 Servei WEB per demanar i controlar continguts de media

Aquest servei WEB s'anomena 'ApiTranscodeModule11', i la seva descripció s'especifica mitjançant un arxiu WSDL[6] (WEB Service Description Language), habilitat a la següent URL:

http://84.88.32.17:8080/Axis/Services/ApiTranscodeModule11?wsdl

Aquest arxiu basat en XML descriu tant els mètodes com els paràmetres i tipus de dades d'entrada i sortida de les funcions implementades pel mòdul. A partir d'aquest arxiu es pot implementar un client per a accedir a les funcions ofertes sense tenir coneixement de la implementació real. Els serveis WEB permeten fer invocació remota de funcions, en aquest cas sobre HTTP/SOAP.

Per a dur a terme el desenvolupament dels serveis WEB, es va fer servir el kit de desenvolupament d'Apache anomenat AXIS, concretament la versió 1.3.

Mitjançant aquesta interfície, el client del servei WEB és capaç de demanar la reproducció d'un vídeo segons unes característiques específiques (procés de transcodificació), controlar el vídeo (stop, play, pause), demanar llistat de continguts disponibles al mòdul, consultar disponibilitat de continguts, ... És a dir, pot accedir a totes les operacions que permet l'API de control desenvolupada en el mòdul.

A continuació és nombraran els diferents mètodes que es van implementar així com la funcionalitat que tenen per descartar directament els que no serveixin pel projecte XAC:

- *createtranscoding*: Aquest mètode crea un nou transcodificador i engega un procés de transcodificació en temps real amb els paràmetres especificats → Aquest mètode no es necessitarà.
- *createTXMediaServer*: Aquest mètode crea un nou servidor (processador de media) en mode transmissió del recurs de media especificat → Aquest mètode s'aprofitarà directament, ja que proporciona el mètode principal per fer un servei d'Streaming.
- *control*: Aquest mètode actua amb un servidor de media realitzant una operació especificada (play, stop, pause...) → Aquest mètode també s'aprofitarà ja que permetrà fer un control dels vídeos que els usuaris estiguin veien en temps real.

[6] Per més informació: http://www.w3.org/TR/wsdl

- *getListTXMediaServers*: Aquest mètode proporciona un llistat dels processadors de media (només els que estan actuant en mode Streaming) que s'han instanciat en el mòdul de transcodificació i que estan transmetent fluxos de media → Aquest mètode no es imprescindible, però podria donar valor afegit.
- *getListMediaTranscodings*: Aquest mètode proporciona un llistat dels mediaProcessors (en mode Transcodificació) registrats en el mòdul de transcodificació que estan transcodificant en temps real (amb unes opcions de transcodificació determinades). El resultat d'aquesta transcodificació en temps real s'està transmetent a un destí identificat per la seva ip i port on espera "rebre el flux" de media → Aquest mètode no es necessitarà per aquesta part del projecte XAC.
- *getListofServers*: Aquest mètode proporciona un llistat de tots dels processadors de media actius que hi ha actius en el mòdul de transcodificació. Inclou els processadors de media que actuïn en mode transcodificació (en temps real), en mode distribució o emmagatzemat) → Aquest mètode tampoc es necessitarà en aquesta part.
- *getResources*: Aquest mètode proporciona una llista de recursos → Aquest mètode no farà falta per al projecte XAC, per tant no s' aprofitarà.

2.2.2.2 Servei WEB per la publicació de continguts

El mòdul de transcodificació *online* permet que els usuaris transfereixin els fitxers de media que volen publicar per a posteriorment transcodificar.

La publicació de continguts al mòdul de transcodificació online es fa a través d'un servei WEB anomenat "UploadService". La transferència del fitxer es fa mitjançant HTTP/SOAP. Un cop que l'usuari publica un contingut en el mòdul de transcodificació online mitjançant el servei WEB, es retorna una referència que identifica el recurs, per a la posterior reproducció d'aquest.

La especificació WSDL està habilitat a la següent URL:

http://84.88.32.17:8080/Axis/Services/UploadService?wsdl

Una altre cosa a tindre en compte, i que ja s'ha vist a l'apartat 1.3, actualment el servei de publicació de fitxers del PI s'ha enfocat cap a una publicació dins d'una xarxa distribuïda basada en el projecte JXTA.

Dintre d'aquesta part només hi ha un mètode que implementarà el servei d'Upload:

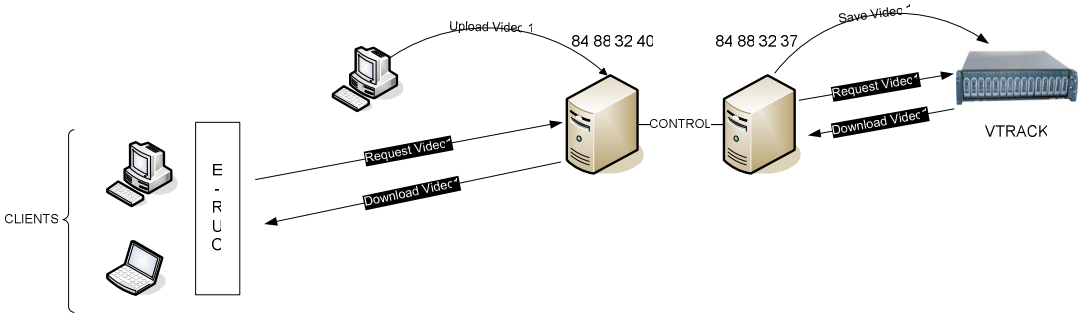
- *publishResource*: Aquest mètode publica un nou recurs de media en el sistema de transcodificació.

Aquesta segona part del PI és interessant ja que és exactament una de les parts que demanen a la XAC: càrrega d'un recurs, per tant si que s'utilitzarà.

Pel que fa a l'Streaming del recurs, seran els mòduls de l'apartat anterior els que s'aprofitaran, però adaptant-los a les especificacions de la XAC, eliminant els mètodes que no serveixin. La implementació i posada en marxa de totes dues funcionalitats és veuran al capítol 3.1 en detall.

2.2.2.3 Esquema d'aquests dos serveis per al projecte XAC

Inicialment (i abans de que arribés l'equip propi del projecte XAC), el que s'ha volgut fer és duplicar tot el servei que ja està en marxa del PI a la mateixa màquina i a partir d'aquí crear un nou servei en un port diferent del Tomcat i eliminant els mètodes i serveis que no facin falta, però a l'hora de fer-ho el fet de tenir dos Tomcats en la mateixa màquina, va portar problemes al mòdul del PI i es va decidir de canviar-la a un altre per a no donés problemes. L'esquema que es va muntar va ser el semblant al del PI (Il·lustració 8).



Il·lustració 10. Esquema XAC amb accés als vídeos del Vtrack

En aquest esquema, els usuaris ja tenen un client e-Ruc (o el portal WEB) amb el que es comuniquen amb el WEB Service per pujar o demanar un vídeo. Aquest WEB Service es comunica amb l'equip que te el servidor VideoLAN el qual fa de traductor de la petició de l'usuari i retorna el vídeo demanat en el cas d'una petició en Streaming o simplement el WEB service guarda el vídeo al vtrack en el cas d'una petició de publicació del contingut (Upload).

Quan, finalment, va arribar l'equip de la XAC es va configurar la IP fixa: 84.88.32.44 i es va duplicar la implementació del PI. Una vegada duplicat el WEB Service, es varen eliminar els mètodes que no feien falta en aquest projecte per deixar estable aquesta part.

Pel que fa al servidor VideoLAN es muntarà també en aquest servidor l'Api que farà de traductor i connexió amb els vídeos igual que està muntat al PI.

2.3 Disseny de les especificacions específiques de la XAC

Una altre part important d'aquest WEB Service i que no estava implementada al PI és el servei Download d'un determinat vídeo del servidor. Per fer-ho, s'ha

continuat amb la filosofia del servei d'Streaming però en comptes de veure'l en temps real, que es descarregués tot el fitxer. A continuació es detallarà el disseny d'aquest servei per fer la implementació al capítol següent, 3.1.

2.3.1 Servei de descàrrega de continguts: Download

A més dels serveis d'Streaming i de càrrega d'un fitxer al servidor, s'afegeix el servei de descàrrega d'un contingut ja referenciat.

La descàrrega del contingut d'aquest mòdul es fa a través d'un servei anomenat "DownloadService". La transferència del fitxer es fa mitjançant HTTP/SOAP.

La especificació també està habilitat a la següent URL:

http://84.88.32.44:8080/Axis/Services/DownloadService?wsdl

Per fer el download d'un contingut s'ha pensat que la forma més fàcil d'implementar-ho és passant-li directament el vídeo com un objecte, per a que el client ho rebi sencer i ho pugui tractar de forma més eficient. Per aquest servei s'ha implementat un únic mètode anomenat *downloadResource* que, després de fer la correcta autenticació del token, verifica que l'arxiu seleccionat existeix i el retorna com a objecte.

2.3.2 Referència dels continguts de vídeo

Per aquest projecte, al contrari que pel PI, s'ha especificat un punt que no s'havia considerat abans: el pas de la demanda d'un fitxer amb un *path* únic. Dintre del PI es van implementar dos scripts que classificaven els vídeos a diferents carpetes segons les característiques dels vídeos (mida, format, duració....).

En aquest cas, però, tots els arxius es trobaran a una mateixa carpeta (aquest punt es veurà a l'apartat 2.3.2), per tant únicament serà necessari demanar el nom del vídeo, i un identificador que serà únic per cada contingut. Aquest identificador serà un número aleatori que s'associarà al vídeo quan es catalogui (la catalogació dels vídeos és un mòdul del projecte XAC, no d'aquest projecte, que permetrà pujar els continguts multimèdia associant-los unes metadades estandarditzades per aquest projecte, que facilitarà la cerca del recurs en la xarxa, i aquest identificador).

Per tant, en tots tres serveis s'especificarà directament el lloc on estan els vídeos, i no serà necessari que el client tingui que buscar el *path*.

2.4 Funcionament dels serveis implementats

A continuació es detalla el funcionament dels tres serveis que s'aprofitaran (adaptant-los a les necessitats del projecte XAC) o s'implementaran per aquest projecte.

2.4.1 Servei d'"Streaming"

Per al servei d'Streaming, s'han aprofitat 3 mètodes dels implementats al PI, adaptant-los a les especificacions del projecte XAC. Els mètodes són: *createTXMediaServer, control i getListTXMediaServers*.

A continuació es descriurà amb més detall cadascun d'aquest mètodes:

1. En primer lloc, el mètode *createTXMediaServer* ens permet crear una instància al servidor VideoLAN, per servir el vídeo en Streaming. La comanda per fer la crida a aquest mètode és la següent:

String createTXMediaServer(String serverType, String output, String input, String protoTx, String token, String signature)

Els paràmetres d'entrada que necessita aquesta comanda són:

- serverType: Ex: vlc, dvts, ultra, etc.
- output: IP i port destí en el següent format. Ex: 147.83.113.11:1234
- input: aquest camp pot ser de 3 tipus segons el flux que es vulgui rebre:
  - network:<ipsource:port>
  - dvb:<Multiplex:channel>
  - file:<filename>
- protoTx: Protocol de transmissió. Ex: udp, rtp.
- token
- signature

La sortida d'aquest mètode ens retorna un XML amb el resultat de la operació. Retorna el valor 200 si s'ha pogut realitzar el procés d'enviament del flux i retorna un codi d'error si hi ha hagut algun error amb una descripció d'aquest.

Per a que aquest mètode es pugui iniciar, s'ha de passar abans per l'autenticació del *token* que s'envia (aquest *token* el dona l'aplicació a l'inici i te un clau associada) i que es fa contra el servidor d'autenticació que ha implementat un altre grup del projecte XAC. Si l'autenticació es correcta, el mètode s'executa normalment, sinó, es retorna un error d'autenticació. Això anirà lligat a tots els mètodes que s'implementin ja que aquest projecte està dissenyat per a uns usuaris que han d'estar prèviament registrats, per tant farà falta aquesta part d'autenticació.

2. Amb el mètode *control* es pot controlar el flux de vídeo que estigui en aquell moment en funcionament. Com que aquest projecte estarà lligat a una interfície gràfica per donar facilitats a l'usuari final, aquest mateix podrà demanar aturar el flux, ficar-ho en pausa o donar-li al play. Aquest mètode s'encarregarà de realitzar aquestes peticions:

```
int control(String serverType, String output, String control, String token, String signature)
```

Els paràmetres que necessita aquest mètode són:

- serverType: Ex: vlc, dvts, ultra, etc.
- output : IP i port destí en el següent format: 147.83.113.11:1234
- control:
  - play
  - stop
  - pause

El resultat d'aquest mètode és igual que l'anterior, si tot funciona correctament, el mètode retorna un XML amb el codi 200, si ha hagut algun problema, el mètode retorna el codi d'error i un String associat a l'error.

3. Finalment, el mètode *getListTXMediaServers* proporciona un llistat dels processadors de media (només els que estan actuant en mode Streaming) que s'han instanciat amb el mètode *createTXMediaServer* i que estan transmetent fluxos de media. Aquest mètode només necessita com a paràmetres el token i la signatura per autenticar-se, ja que retornarà un llistat dels servidors de media que trobi.

```
String getListTXMediaServers (String token, String signature);
```

El format de la sortida del mètode serà també un XML i tindrà el següent format:

```
200:response

String response=
<Response>
<numApis>1</numApis><api id="0">
   <mediaServers>
   <mediaServer id="s1" type="vlc" ip="x" port="p"
    resource="gaudi2" format="mp4v" proto="rtp"/>
   <mediaServer  id="s2" type="dvts" ip="x" port="p"
    resource="gaudi2" format="mp4v" proto="rtp"/>
   <mediaServer  id="s3" type="dvts" />
     </api>
   </mediaServers>
</Response>
```

2.4.2 Servei d'"Upload"

En el cas del servei d'Upload únicament s'utilitzarà el mètode implementat per aquesta part: *PublishResource*.

```
String publishResource(DataHandler dh, String filename, String token, String signature )
```

Amb aquest mètode es publica un nou recurs dintre del sistema, és a dir, es puja un nou contingut de vídeo. Els paràmetres d'entrada que necessita aquest mètode són:

- Dh: fitxer a emmagatzemar
- Filename: nom del fitxer per a emmagatzemar
- Token
- Signature

Amb aquest mètode es torna a tenir el mateix format de respostes: es retorna un XML amb el resultat de la operació, es retorna el valor 200 si s'ha pogut realitzar el procés de publicació correctament i es retorna codi d'error si hi ha hagut algun error amb la descripció d' aquest.

Cal comentar en aquesta part, que la implementació de tots dos serveis anteriors, s'ha duplicat des del PI per al projecte XAC i que només s'han fet modificacions per a que pogués funcionar amb les llibreries de la XAC en comptes de les del PI, i que s'ha eliminat part dels codis que no eren necessaris per aquest projecte.

2.4.3 Servei de "Download

El servei de Download inicialment no estava implementat al PI, per tant era una part pròpia del projecte XAC. Per aquest servei s'ha implementat un únic mètode anomenat *downloadResource*:

```
public Object downloadResource(String reference, String token, String signature)
```

Amb aquest mètode, el paràmetre d'entrada més important és la referència a l'arxiu que es vol descarregar:

- reference: on s'indica la referència del contingut prèviament publicat i catalogat

Amb la crida d'aquest mètode, la resposta que retorna és en funció de si l'operació ha estat correcta o no, exactament igual als altres serveis:

- La funció retorna un XML amb el resultat de l'operació. Retorna el fitxer si s'ha pogut realitzar el procés de descàrrega. Retorna codi error si ha hagut algun error amb una descripció d'aquest:
  - Possibles errors:
    - Fitxer sense permís de lectura
    - Fitxer no existeix
    - Oblit d'indicar nom de fitxer o algun dels altres paràmetres
    - Altres excepcions

2.5 Estabilitat de la implementació

Com qualsevol servei que es planifica per donar a un numero determinat d'usuaris o clients finals ha de tindre certa estabilitat. En una primera instància, i com ja s'ha anat comentant, aquest projecte és una prova pilot, i per tant l'estabilitat d'aquest mòdul no és crítica. S'ha de tindre en compte però, que la finalitat d'aquest projecte és de crear una arquitectura de forma estable, i serà, per tant, necessari tindre en compte diverses consideracions per acomplir-ho. Als següents apartats s'explicaran dos temes necessaris per a aconseguir una mínima estabilitat.

2.5.1 Redundància d'equips

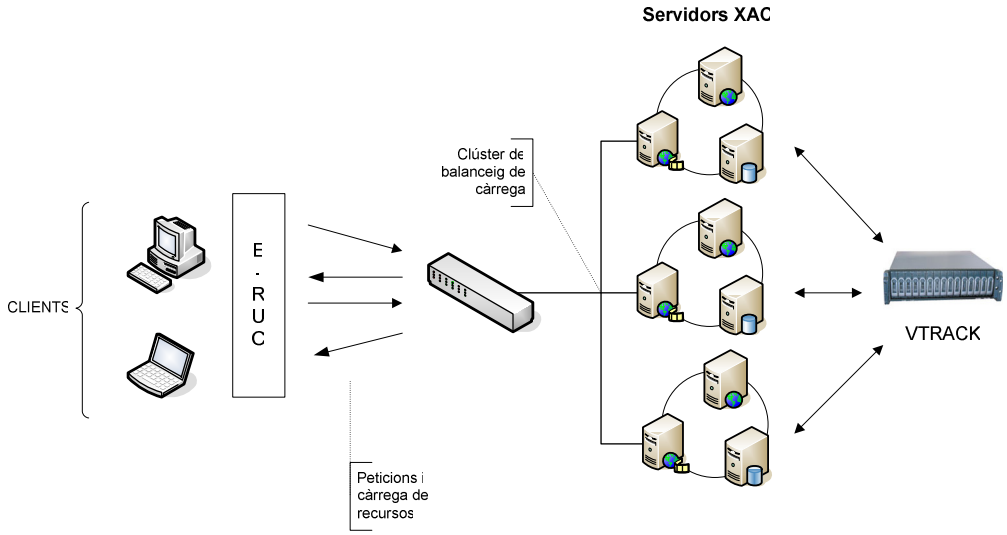
Com ja s'ha vist, aquest projecte està basat en diferents mòduls coordinats per diferents grups que implementen les seves parts en diferents WEB Services. Aquesta solució és provisional ja que la idea és d'unir tot el servei en dos equips per a que estiguin tots els mòduls al mateix lloc. Aquesta solució ja aportaria estabilitat ja que no s'hauria de dependre en cas d'error de cada un dels grups sinó que només un l'hauria de gestionar.

També s'ha vist que aquest projecte tractarà amb molts vídeos que podran emmagatzemar-se al mateix equip local de l'usuari distribuint-lo a través de la xarxa P2P o també a un servidor remot (en principi els discos del vtrack del MediaCAT). Per a que aquesta arquitectura estigui estable el màxim de temps possible, es fa necessari replicar equips que facin de servidor tant dels vídeos com dels diferents mòduls que hi ha al projecte.

Seria important per tant, el fet de tindre un equip duplicat que, per exemple, accepti les peticions dels diferents vídeos, així com un equip duplicat que gestioni els usuaris (les base de dades) i els permisos per accedir als recursos. Aquest tema és important ja que si, per exemple, només existeix un equip amb tots els mòduls i aquest equip cau, el servei deixarà de funcionar.

2.5.1.1 Clúster de balanceig

L'idea a mig termini que s'hauria d'aplicar és la de fer un clúster de balanceig de càrrega on els diferents equips que formin el clúster (veure [5]) continguin tota la informació necessària per poder oferir el servei als diferents usuaris.



Il·lustració 11. Esquema del projecte amb el clúster de balanceig de càrrega

A la imatge es mostra diferents grups de servidors on s'emmagatzemen els mòduls del projecte (podrien ser una, dos, tres màquines...) i aquests servidors duplicats dues vegades més. Aquest fet faria del sistema una arquitectura molt més escalable on el número d'usuaris que podrien interactuar alhora seria major. Aquesta arquitectura també podria proporcionar major seguretat en front d'errors d'algun equip (tenint en compte que no només s'haurien de duplicar els serveis WEB sinó també el servidor VideoLAN per a servir els vídeos en Streaming).

Un aspecte important a considerar en aquest esquema és la ubicació dels diferents grups de servidors ja que si tots es troben al mateix lloc, un error de xarxa o d'electricitat faria que cap grup pogués funcionar, per tant és important situar cada grup a un lloc diferent on es pugui portar un control.

Al capítol d'implementació, al 3.2 concretament, s'explicarà una manera de fer aquest balanceig de forma fàcil, ràpida i eficient a curt termini.

2.5.2 Gestió de la implementació

Un altre aspecte important per donar més estabilitat al sistema és crear un bon sistema de gestió. El tema de la gestió de xarxes o equips és molt ampli i existeixen moltes solucions actualment que permeten el bon funcionament dels serveis.

En aquest projecte, si s'aconseguís fer una estructura amb balanceig de càrrega, aquest sistema ja proporcionaria una estabilitat àmplia, que es podria complementar amb un control dels diferents grups de servidors, ja sigui controlant una a una l'estat de les màquines, o l'estat de la xarxa (número d'usuaris, vídeos en petició, estat dels discos (vtrack), eficiència...).

Les eines que es troben al mercat per a fer una gestió dels equips o de la xarxa són molt diverses, però centrades als mateixos aspectes.

Inicialment podríem distingir el fer-ne una gestió de xarxa o a nivell d'equip. Per al cas del projecte XAC, no serà tan crític el tindre una monitorització contínua de la xarxa (ja que com ja s'ha comentat, la plataforma on estarà la maqueta del projecte està contínuament amb proves), però sí a nivell d'equip i de serveis o ports (del Tomcat per exemple).

2.5.2.1 Proposta

Una bona proposta de gestió hauria de disposar d'una sèrie de recursos que inicialment, a una plataforma en proves, no es disposa. La idea llavors és aprofitar les eines que ja s'utilitzen i poder fer una gestió del sistema, tot i que no sigui per una estabilitat del 100%.

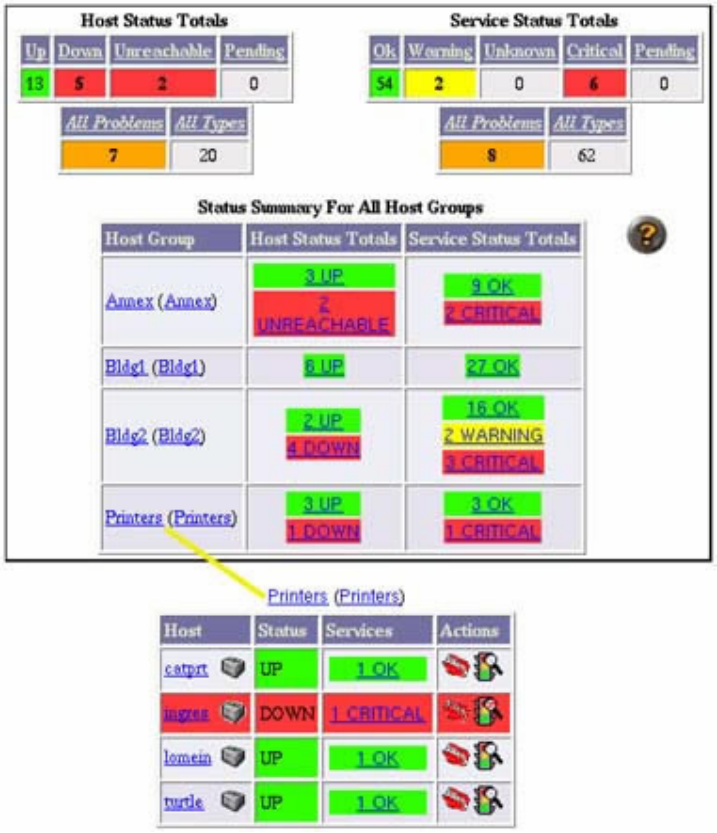
Per fer-ho, s'ha pensat en la utilització del software Nagios[7] que actualment ja s'utilitza a la xarxa d'i2CAT. Aquest software de lliure distribució ens permet controlar una sèrie de paràmetres o serveis d'un equip i actuar ràpidament davant d'un problema. El fet de tenir només un parell d'equips en el sistema, facilita també la gestió que s'ha de fer ja que només s'hauran de controlar aquests equips.

Amb aquesta solució s'aconsegueix un mínim control del sistema, monitoritzant paràmetres com CPU o memòria de l'equip com ports o serveis actius a cada màquina.

En el cas que fos viable actualment fer l'arquitectura proposta amb balanceig de càrrega, una bona gestió de tot el sistema proporcionaria una seguretat i estabilitat molt més completa.

A la següent imatge es representa un sistema d'exemple on es pot veure com mostra Nagios l'estat de la xarxa:

[7] http://www.nagios.org/



Il·lustració 12. Exemple estat de la xarxa amb Nagios

Amb aquesta monitorització que es pot seguir via WEB es pot controlar que equips o serveis funcionin correctament i veure si tenen algun problema. També es pot fer de manera automàtica, que s'actuï executant alguna comanda o arxiu per tal de reiniciar un servei o fins i tot el propi equip, quan hi ha algun problema.

Amb aquestes dues propostes, balanceig de càrrega i gestió, s'aconsegueix un sistema més estable, amb menys errors i més controlats, on també pot donar servei a un major numero d'usuaris a la vegada. També al capítol següent, concretament a l'apartat 3.3 s'implementarà un sistema de gestió inicial utilitzant el Nagios de MediaCAT.

3 Capítol 3. Implementació

3.1 Creació dels tres serveis WEB per la XAC

A continuació s'explicarà, pas a pas, com s'ha implementat tot el disseny explicat al capítol anterior per als tres serveis: Upload, Download i Streaming.

- Primer s'ha de buscar una màquina que farà de WEB Services d'aquest mòdul del projecte XAC. L'equip esta configurat amb la IP 84.88.32.44.
- Es duplica llavors el servei corresponen al mòdul de transcodificació del PI:

    - Aquest servei es troba a un equip servidor amb Debian 2.4.27 amb un servidor Jakarta-Tomcat mentre que l'equip que s'utilitzarà te una Debian 2.6.8.
    - El directori del PI que s'ha de duplicar es troba a:
      Directori: */usr/local/jakarta-tomcat-4.1.31/*
    - Dintre d'aquest directori trobem tant les configuracions pròpies de l'Apache-Axis com el codi i les classes corresponents a l'aplicació del PI per al dos serveis explicats a l'apartat 2.2.2.1 i 2.2.2.2 (al PI es troben el serveis Streaming i Upload, el Download serà una especificació pròpia de la XAC):
      /usr/local/jakarta-tomcat-4.1.31/bin → stop/start Tomcat
      /usr/local/jakarta-tomcat-4.1.31/Webapps/Axis → codis java i configuracions del projecte

A l'apartat de disseny de l'Streaming i del Upload, s'han vist els mètodes que es necessitaran i el que no, per tant s'eliminen, primer de tot, aquest mètodes.

Per a explicar com ficar en marxa aquest nou WEB Service es farà pas a pas el seguiment que s'ha portat a terme per crear cadascun dels serveis (tant per l'Updoad, com pel Download o per l'Streaming s'ha de seguir el mateix procediment).

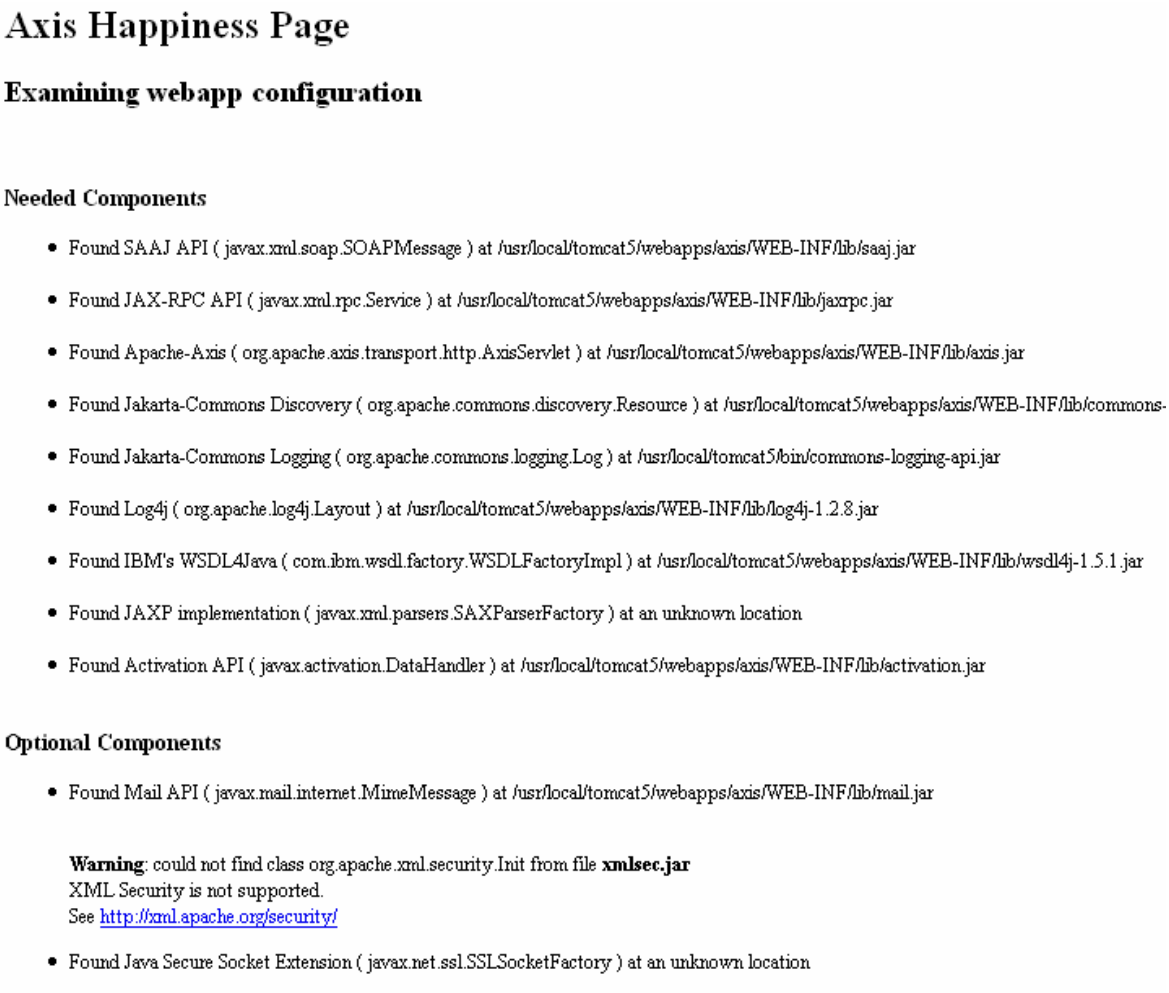
3.1.1 Passos per la creació dels serveis WEB

El primer que s'ha de preparar és l'entorn de treball, instal·lant els programes adequats i necessaris. Aquest software s'ha comentat a l'apartat del disseny 2.2.

3.1.1.1 Instal·lació de Apache Axis 1.3

1) En primer lloc s'ha d'instal·lar i configurar el servidor d'aplicacions Apache Tomcat (per descarregar-lo, veure [6]).

2) Descarregar en segon lloc els binaris que corresponen a la instal·lació d'Axis. Dintre d'aquesta distribució es troba la carpeta *Webapps/Axis*, que ha d'anar dintre de la carpeta del servidor Tomcat (*/usr/local/tomcat5/webapps/axis*)
3) Una vegada descarregat i amb els binaris d'axis a la carpeta corresponen, ja es pot accedir a la pàgina principal:
   http://84.88.32.44:8080/Axis
4) Des d'aquesta pàgina es pot comprovar la instal·lació fent clic al link *validate*. Aquesta pàgina ens mostrarà les llibreries necessàries per a que Axis pugui funcionar. Fins que aquesta validació no es doni per bona, no es pot continuar ja que Axis no funcionarà fins que no tingui totes les llibreries obligatòries instal·lades.
   Les llibreries s'han d'afegir a la carpeta */tomcat5/Webapps/Axis/WEB-INF/lib.* A la següent imatge es mostra un tros de la pàgina amb la validació d'Axis.



Axis Happiness Page

Examining webapp configuration

Needed Components

- Found SAAJ API (javax.xml.soap.SOAPMessage) at /usr/local/tomcat5/webapps/axis/WEB-INF/lib/saaj.jar
- Found JAX-RPC API (javax.xml.rpc.Service) at /usr/local/tomcat5/webapps/axis/WEB-INF/lib/jaxrpc.jar
- Found Apache-Axis (org.apache.axis.transport.http.AxisServlet) at /usr/local/tomcat5/webapps/axis/WEB-INF/lib/axis.jar
- Found Jakarta-Commons Discovery (org.apache.commons.discovery.Resource) at /usr/local/tomcat5/webapps/axis/WEB-INF/lib/commons-
- Found Jakarta-Commons Logging (org.apache.commons.logging.Log) at /usr/local/tomcat5/bin/commons-logging-api.jar
- Found Log4j (org.apache.log4j.Layout) at /usr/local/tomcat5/webapps/axis/WEB-INF/lib/log4j-1.2.8.jar
- Found IBM's WSDL4Java (com.ibm.wsdl.factory.WSDLFactoryImpl) at /usr/local/tomcat5/webapps/axis/WEB-INF/lib/wsdl4j-1.5.1.jar
- Found JAXP implementation (javax.xml.parsers.SAXParserFactory) at an unknown location
- Found Activation API (javax.activation.DataHandler) at /usr/local/tomcat5/webapps/axis/WEB-INF/lib/activation.jar

Optional Components

- Found Mail API (javax.mail.internet.MimeMessage) at /usr/local/tomcat5/webapps/axis/WEB-INF/lib/mail.jar

  Warning: could not find class org.apache.xml.security.Init from file **xmlsec.jar**
  XML Security is not supported.
  See http://xml.apache.org/security/

- Found Java Secure Socket Extension (javax.net.ssl.SSLSocketFactory) at an unknown location

Il·lustració 13. Validació de la instal·lació d'Axis

3.1.1.2 Preparació de l'entorn de treball

Per poder compilar i desplegar els serveis WEB que es crearan, és important configurar correctament les variables d'entorn tant per Java com per Axis per a que el sistema tingui accés a les llibreries i al path de cadascun.

```
set JAVA_HOME= /root/jdk1.5.0_06
set PATH=$PATH:/ root/jdk1.5.0_06/bin

set AXIS_HOME= /usr/local/tomcat5/Webapps/Axis
set AXIS_LIB=$AXIS_HOME/lib
set CATALINA_HOME="/usr/local/tomcat5/

CLASSPATH=JAVA_HOME/lib:/usr/local/tomcat5/Webapps/Axis/WEB-
INF/lib/Axis.jar:/usr/local/tomcat5/Webapps/Axis/WEB-
INF/lib/mail.jar:/usr/local/tomcat5/Webapps/Axis/WEB-
INF/lib/activation.jar:/usr/local/tomcat5/Webapps/Axis/WEB-INF/lib/i2cat-sec-1-
4.jar:/usr/local/tomcat5/Webapps/Axis/WEB-INF/lib/Axis-
ant.jar:/usr/local/tomcat5/Webapps/Axis/WEB-
INF/lib/jaxrpc.jar:/usr/local/tomcat5/Webapps/Axis/WEB-INF/lib/commons-
discovery.jar:/usr/local/tomcat5/Webapps/Axis/WEB-INF/lib/commons-
logging.jar:/usr/local/tomcat5/Webapps/Axis/WEB-
INF/lib/wsdl4j.jar:/usr/local/tomcat5/Webapps/Axis/WEB-INF/lib/log4j-
1.2.8.jar:/usr/local/tomcat5/Webapps/Axis/WEB-INF/lib/saaj.jar:

export PATH JAVA_HOME CLASSPATH AXIS_LIB AXIS_HOME CATALINA_HOME
```

3.1.1.3 Creació del serveis WEB

Una vegada es té tot l'entorn preparat, passem a crear el servei WEB. Per fer-ho, s'ha de seguir 4 passos:

1. Implementar el codi de les operacions que es volen fer (en aquest cas el Upload/Download i l'Streaming de continguts)
2. Obtenir l'arxiu WSDL del codi implementat per a facilitar la integració d'altre mòduls amb aquest i adaptar-ho també als altres mòduls del projecte XAC a través del seu codi WSDL.
3. Obtenir el package dels codis Java per a obtenir les interfícies amb el WSDL2Java.
4. Desplegar el servei WEB

Com que són tres serveis els que es volen implementar en aquesta part, s'han de seguir aquest quatre passos anteriors (que s'expliquen a continuació) per tots tres serveis per separat. S'explicarà llavors un dels serveis a mode d'exemple, sent els altres dos exactament igual (canviant els codis Java...).

3.1.1.3.1 Implementació del codi en Java

El primer pas que s'ha de fer és implementar el codi en Java del servei que es vol fer. Per al cas de l'Streaming i el Upload, com ja s'ha vist, s'aprofita el codi del PI i per tant el que s'ha de fer és eliminar els mètodes que no s'utilitzaran

(especificats a l'apartat 2.2.2). El nom dels serveis seran ApiTranscodeModule i UploadService i els mètodes:

- Per al cas de l'Streaming s'utilitzaran els mètodes (ApiTranscodeModule):
    - control
    - createTXMediaServer
    - getListTXMediaServers
- Per a l'UpdoadService s'utilitzarà la mateixa implementació que per al PI:
    - publishResource
- Per al cas del Download, s'implementa el codi en Java partint des de zero, i el mètode serà:
    - DownloadResource

Llavors es compila el codi de nou per obtenir les classes correctes per a cada cas.

3.1.1.3.2 Obtenir del codi en Java el WSDL

Es crearà l'arxiu WSDL a partir del codi de la implementació que s'ha fet, a partir de l'eina Axis Java2WSDL[8] (veure [7]). La informació que necessita aquesta eina és (exemple del cas del servei d'Streaming):

- Nom del fitxer WSDL (ApiTranscodeModule.wsdl)
- URL del servei WEB (http://localhost:8080/Axis/Services/ApiTranscodeModule)
- Target namespace (urn:ApiTranscodeModule)
- Mapeig del package de Java amb el namespace (ApiTranscodeModule = urn:ApiTranscodeModule)
- Nom de la classe (ApiTranscodeModule)

Per executar aquesta eina llavors, s'ha d'utilitzar la comanda següent:

```
/usr/local/tomcat5/Webapps/Axis#java org.apache.Axis.wsdl.Java2WSDL –o ApiTranscodeModule.wsdl –lhttp://localhost:8080/Axis/Services/ApiTranscodeModule –n urn:ApiTranscodeModule –p "ApiTranscodeModule" urn:ApiTranscodeModule ApiTranscodeModule
```

Amb aquesta comanda s'obté el WSDL corresponen al servei WEB i que es podrà consultar via WEB a la següents adreces (una per cada servei), a més a més s'adjunten aquests WSDL al 8.2, Annex B :

[8] Per més informació: http://ws.apache.org/Axis/java/user-guide.html o http://www.onjava.com/pub/a/onjava/2002/06/05/Axis.html?page=2

http://84.88.32.44:8080/Axis/Services/ApiTranscodeModule?wsdl –> Streaming
http://84.88.32.44:8080/Axis/Services/UploadService?wsdl → Upload
http://84.88.32.44:8080/Axis/Services/DownloadService?wsdl → Download

3.1.1.3.3 Utilització del WSDL per a crear els stubs de comunicació per als altres mòduls de la XAC: WSDL2Java

Amb l'eina d'Axis WSDL2Java es podrà generar el codi que permetrà desplegar el servei i crear els stubs de comunicació per als altres mòduls.

Continuant amb el servei d'Streaming es generarà el package ApiTranscodeModule_pkg. El WSDL2Java requereix els següents paràmetres:

- Directori base de destí (*-o* .)
- Desplegament del servei (aplicació, petició o sessió: *-d Application*)
- Habilitar la generació del server-side (en cas que no accedim a un WEB service extern, sinó amb el client no faria falta: *-s*)
- Package on es ficarà el codi (*-p ApiTranscodeModule_pkg*)
- Nom del fitxer WSDL (*ApiTranscodeModule.wsdl*)

Per tant, la comanda a executar és:

/usr/local/tomcat5/Webapps/Axis#java org.apache.Axis.wsdl.WSDL2JAVA –server-side –skeletonDeploy true –p ApiTranscodeModule_pkg ApiTranscodeModule.wsdl

Amb l'execució d'aquesta comanda es generen els següents arxius:

- *ApiTranscodeModuleSoapBindingImpl.java*: és la implementació del codi per al WEB service. Aquest arxiu s'ha d'editar per a lligar-lo a la implementació realitzada.
- *ApiTranscodeModule11.java*: interfície remota per a l'aplicació.
- *ApiTranscodeModule11Service.java*: Interfície de servei del WEB Service.
- *ApiTranscodeModule11ServiceLocator.java*: localització del servei.
- *ApiTranscodeModuleSoapBindingSkeleton.java*: codi amb l'skeleton del servidor.
- *ApiTranscodeModuleSoapBindingStub.java*: codi amb l'stub del client que encapsula l'accés del client.
- *Deploy.wsdd*: descriptor que anuncia al sistema Axis pel desplegament del servei.
- *Undeploy.wsdd*: descriptor que anuncia al sistema Axis per a que despubliqui el servei.

S'ha de modificar llavors l'arxiu ApiTranscodeModuleSoapBindingImpl.java per lligar-ho al servei WEB, ja que per defecte aquesta classe retorna nul per tots els mètodes.

- Primer es crea una instància a la classe principal :

Import ApiTranscodeModule_pkg.ApiTranscodeModule11

```
ApiTranscodeModule11 atm;
```

- I llavors, per cada mètode es fa la crida a aquesta classe i es retorna el propi mètode:

```
atm=new ApiTranscodeModule();
return atm.nomdelmetode(paràmetres que rep el mètode);
```

Una vegada editat correctament aquest fitxer, es compila tot el codi per obtenir les classes corresponents.

Aquest pas s'ha de fer també per a cada servei. Es tindrà llavors un package per a cada servei que contindrà els stubs de comunicació per als altres mòduls del projecte i un arxiu deploy.wsdd/undeploy.wsdd que s'encarregarà de desplegar o despublicar el servei WEB.

3.1.1.3.4 Desplegar el Servei WEB

L'últim pas per a la creació del WEB Service és fer el desplegament del servei sobre el servidor que s'ha instal·lat, en aquest cas Tomcat 5.5.

En primer lloc s'ha de crear el .jar amb el codi implementat (s'ha de tindre en compte que s'han d'afegir tots els package que es necessiten pel servei: autenticació del token, creació de les sessions...):

```
jar cvf ApiTranscodeModule11.jar pApiTranscodeModule11/*.class
pApiTranscodeModule11/AutenticadorXAC/*.class pApiTranscodeModule11/model/*.class
pApiTranscodeModule11/pBalanceador/*.class pApiTranscodeModule11/pTreeView/*.class
pApiTranscodeModule11/session/*.class ApiTranscodeModule11_pkg/*.class
```

Es copia llavors aquest jar (ApiTranscodeModule11.jar) dintre de la carpeta WEB-INF/lib

```
mv ApiTranscodeModule11.jar tomcat5/Webapps/Axis/WEB-INF/lib/.
```

Si tot es correcte apareixerà el missatge següent:

```
Processing file pApiTranscodeModule11/deploy.wsdd
<Admin>Done Processing</Admin>
```

Si s'accedeix llavors a la pàgina comentada anteriorment(3.1.1.3.2), es comprovarà si el servei s'ha desplegat correctament.

Una vegada desplegat el servei corresponen a l'Streaming (ApiTranscodeModule), es segueixen els mateixos passos pels altres dos serveis, i finalment a la URL del WEB service es comprova que estan publicats tots tres serveis::

Il·lustració 14. Serveis publicats en WEB correctament

3.1.1.4 Api per al servidor VideoLAN

Com s'ha vist al punt 2.2.1, per servir continguts audiovisuals es necessària una connexió amb un servidor de VideoLAN per a que es serveixin els vídeos en Streaming.

L'equip que inicialment s'utilitza és el servidor que es va crear per al PI, tot i que es crearà aquesta mateixa Api VideoLAN a l'equip propi de la XAC per a que es pugui servir tot amb el mateix equip.

L'equip que te l'Api de VideoLAN és el 84.88.32.37, per tant, es crea un arxiu de *properties* on s'estableix aquesta IP com a Api i que es cridarà des de la implementació de l'Streaming.

A l'annex 8.3 es detalla el funcionament d'aquesta Api de control per als servidors VideoLAN.

3.1.1.5 Directori per emmagatzemar els vídeos

Una altre part a tenir en compte, seran els directoris on es guardaran els vídeos que es pugin amb el servei d'Upload implementat. Per a considerar-los dintre del codi, es crea també un arxiu de *properties* (per poder modificar-ho més fàcilment en cas de que canviï) on establim els paths on es guardaran aquests recursos.

Com ja s'ha comentat, al PI hi ha gran varietat de vídeos amb diferents qualitats o formats, pel que va ser necessari crear dos scripts senzills per poder extreure les característiques del vídeo i poder catalogar-lo automàticament a una carpeta. Per al cas del projecte XAC, les especificacions fixen un únic directori per a fer mes senzill la càrrega i descarrega de fitxers. Per tant, es crearà un directori que serà únicament per a emmagatzemar els vídeos d'aquest projecte. Es configuren llavors els scripts que criden els tres serveis amb únic directori que serà:

/home/mediacat

Per tant tots tres serveis accediran a aquest directori per pujar o baixar els vídeos, que quedaran identificats pel seu nom i un identificador únic per cada vídeo i que es farà aleatori per afegir-lo al nom (tal i com s'ha explicat a l'apartat 2.3.2).

3.2 Implementació de l'escalabilitat

En aquesta primera versió de funcionament d'aquest WEB Service, no es tindrà en compte el tema d'escalabilitat ja que no és inicialment el propòsit. Tot i això, es considera que és un tema molt important per al funcionament definitiu d'aquest servei i per tant s'explicarà una manera fàcil de d'implementar-ho.

En cas de tenir els servidors duplicats tal i com s'explicava al disseny, capítol 2.5, es pot fer un balanceig de càrrega entre els servidors Tomcats i també entre els servidors de vídeo (VideoLAN), de la següent manera:

3.2.1 Jakarta-Tomcat: Load Balancer

Per fer el balanceig de Tomcats es necessari de configurar les opcions *mod_proxy* i *mod_rewrite* i el connector AJP (veure [9]). Aquesta opció no es de les que aporten mes avantatges, però es molt fàcil d'instal·lar i configurar.

S'edita l'arxiu */usr/local/tomcat5/conf/server.xml*. Es busca l'etiqueta *Engine* i s'afegeix a tots els casos un nou atribut anomenat *jvmRoute* amb valor *tomcat1, tomcat2, tomcat3*...

```
<Engine name="MainEngine" defaultHost="localhost" jvmRoute="tomcat1">
```

Cadascun d'aquest atributs seran els diferents equips duplicats d'aquest WEB Service. Aleshores es configura a l'Apache l'arxiu worker.properties on s'especifiquen els atributs anteriors lligats a les IPs dels equips, per exemple:

```
#
# workers.properties
#
```

```
# list the workers by name

worker.list=tomcat1, tomcat2, loadbalancer

# ------------------------
# First tomcat server
# ------------------------
worker.tomcat1.port=11009
worker.tomcat1.host=localhost
worker.tomcat1.type=ajp13

# Specify the size of the open connection cache.
#worker.tomcat1.cachesize

#
# Specifies the load balance factor when used with
# a load balancing worker.
# Note:
#  ----> lbfactor must be > 0
#  ----> Low lbfactor means less work done by the worker.
worker.tomcat1.lbfactor=100


# ------------------------
# Second tomcat server
# ------------------------
worker.tomcat2.port=12009
worker.tomcat2.host=localhost
worker.tomcat2.type=ajp13

# Specify the size of the open connection cache.
#worker.tomcat2.cachesize

#
# Specifies the load balance factor when used with
# a load balancing worker.
# Note:
#  ----> lbfactor must be > 0
#  ----> Low lbfactor means less work done by the worker.
worker.tomcat2.lbfactor=100


# ------------------------
# Load Balancer worker
# ------------------------

#
# The loadbalancer (type lb) worker performs weighted round-robin
# load balancing with sticky sessions.
# Note:
#  ----> If a worker dies, the load balancer will check its state
#        once in a while. Until then all work is redirected to peer
#        worker.
worker.loadbalancer.type=lb
worker.loadbalancer.balanced_workers=tomcat1, tomcat2

#
# END workers.properties
#
```

Amb això s'associa cada *worker*, cada equip duplicat, a una determinada IP i un port.

Dintre del mateix Apache2, s'ha de configurar també l'arxiu *httpd.conf*:

```
RewriteMap SERVERS rnd:/usr/local/httpd/conf/servers.xml

<Location /webapp>
   RewriteEngine On

   RewriteCond"%{HTTP_COOKIE}"              "(^|;\s*)jsessionid=\w*\.(\w+)($|;)"
   RewriteRule "(.*)"              "http://${SERVERS:%2}%{REQUEST_URI}"  [P,L]
   RewriteRule "^.*;jsessionid=\w*\.(\w+)($|;)"  "http://${SERVERS:$1}%{REQUEST_URI}"  [P,L]
   RewriteRule "(.*)"                        "http://${SERVERS:ALL}%{REQUEST_URI}" [P,L]
</Location>
```

Amb la primera línia s'especifica que l'arxiu que carreguem es el que s'ha modificat abans, *server.xml*.

Es crea a continuació un arxiu anomenat per exemple *tomcat_jk.conf*, que s'ha d'afegir en $APACHE_HOME/conf/httpd.conf. Aquest arxiu carrega el mòdul *mod_jk*:

```
LoadModule jk_module libexec/mod_jk.so
AddModule mod_jk.c
```

Seguidament es configura l'arxiu *mod_jk*:

```
# Location of the worker file
JkWorkersFile "/etc/httpd/conf/jk/workers.properties"
# Location of the log file
JkLogFile "/etc/httpd/jk/logs/mod_jk.log"
# Log level : debug, info, error or emerg
JkLogLevel emerg
# Assign specific URL to Tomcat workers
JkMount /admin loadbalancer
JkMount /admin/* loadbalancer
JkMount /examples loadbalancer
JkMount /examples/* loadbalancer
```

Amb aquesta configuració s'aconseguiria fer un balanceig entre els dos equips que s'han considerat per l'exemple. Com ja s'ha comentat, no és una de les millors solucions que existeixen per fer el balanceig, però si una solució ràpida i fàcil d'instal·lar que serviria per ficar en marxa el balanceig dels servidors de tomcat a curt termini.

3.2.2 Duplicació del servidor VideoLAN

Pel que fa al servidor de VideoLAN, el fer un balanceig es més fàcil, ja que ja es va considerar aquesta opció al fer tota la implementació.

A la implementació del servei de l'Streaming, es va considerar de ficar en un arxiu de *properties*, anomenat listadoApis, que després s'interpreta al codi, i on

es fica el numero d'Apis de VideoLAN que hi han, i a continuació la IP i el port de cadascuna d'elles. L'arxiu es tan fàcil com:

```
/*listado de apis*/
numApis=1

ip1=84.88.32.44
port1=9912

ip2=84.88.32.37
port2=9912
```

I la part del codi on s'inicialitzen aquestes Apis és:

```
this.fichero = (PropertyResourceBundle) ResourceBundle.getBundle("listadoApis");
this.numApis = Integer.parseInt(fichero.getString("numApis"));
this.apis = new Api[numApis];
String ip;
int port;
for (int i = 0; i < numApis; i++) {
        ip = fichero.getString("ip" + (i + 1));
        port = Integer.parseInt(fichero.getString("port" + (i + 1)));
        apis[i] = new Api(ip, port);
}
```

Amb aquesta mètode es crea un objecte on es guarden totes les Apis que estan a l'arxiu i que després s'utilitzaran per fer el balanceig. A continuació es crea un mètode on es buscarà l'Api que no estigui activa en aquell moment, per poder donar servei:

```
public Api obtenApi() {
   Api aux = null;
   boolean found = false;
   int i = 0;
   while (i < apis.length && found == false) {
     if (apis[i].getDisponibilidad()) {
       aux = apis[i];
       found = true;
     }
     i++;
   }
   return aux;
 }
```

Aquesta opció està disponible des del principi ja que es va dissenyar el codi de l'Streaming ja pensant en donar balanceig, però actualment només hi ha un equip que fa de VideoLAN, per tant, encara no es pot fer cap balanceig.

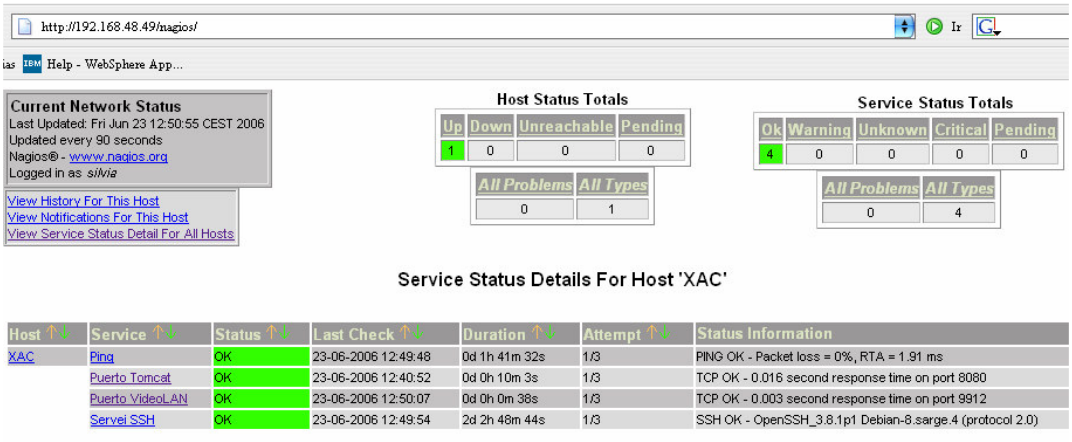
3.3 Implementació de la gestió

Inicialment, la gestió d'aquest mòdul es basarà bàsicament en el que s'ha explicat al capítol 2.5.2, amb el *Nagios*. Per tant, s'analitzaran els serveis o aplicacions que es poden monitoritzar del servidor per després afegir-los.

Al servidor WEB que s'ha muntat, es considera que els serveis més importants que s'han de tindre monitoritzats són:

- La interfície de xarxa: es controlarà que sempre es pugui fer un *ping* al servidor per controlar que no caigui la interfície o que no es perdi la IP
- El port del Tomcat, 8080: es farà un chequeig del port per controlar que no caigui el servidor de Tomcat
- El servei SSH per poder controlar remotament els tres serveis implementats
- El port on es troba el servidor VideoLAN, 9912, per controlar que no es tanqui

Una vegada monitoritzats aquests serveis, el que és fer un arxiu amb un script per cada servei, on es crearà una comanda que reiniciarà aquests daemons quan hi hagi algun problema. L'explicació per configurar correctament el Nagios amb aquest chequejos es troba a l'annex 8.4. Finalment, amb la configuració correctament feta, es mostrarà la gestió via WEB:



Il·lustració 15. Serveis de l'equip de la XAC monitoritzats per Nagios

A la Il·lustració 15 es pot observar que tots quatre serveis monitoritzats funcionen correctament ja que el seu estat es "OK". En el cas que hagués algun problema a algun dels serveis monitoritzats, la mateixa aplicació de Nagios te configurat un paràmetre per enviar un correu al responsable d'aquests equips amb la notificació del error del servei.

Amb aquest servei s'aconsegueix una bona gestió inicial pels serveis implementats pel mòdul del projecte XAC.

4 Capítol 4. Proves

4.1 Proves del servei implementat

Arribat aquest punt, ja es tenen publicats correctament tots tres serveis necessaris per aquest projecte, però encara no s'ha provat si funcionen correctament amb els arxius de *properties*, el servidor de VideoLAN o l'accés als directoris dels vídeos.

En aquest capítol, s'explicaran les proves realitzades al WEB Service per comprovar que funcionen correctament els tres serveis, així com els problemes que han anat apareixent durant la implementació.

4.1.1 Clients per les proves dels serveis reciclats del PI

Per fer les proves inicials amb les implementacions realitzades, s'han creat uns clients que fan una petició directa als serveis implementats (amb una autenticació prèvia al servidor d'autenticació del projecte XAC). Amb aquest clients es pot verificar si els serveis funcionen correctament.

Per provar el servei d'Streaming es crea una instància amb els paràmetres directes per fer la petició:

```
String res;
ApiTranscodeModule11Service servicea = new ApiTranscodeModule11ServiceLocator();
System.out.println(servicea.getApiTranscodeModuleAddress());
ApiTranscodeModule11 porta = servicea.getApiTranscodeModule();

res = porta.createTXMediaServer("vlc", "84.88.32.43:1234", "park_guell.m2t", "udp", tok, "");
//res=porta.control("vlc","84.88.32.43:1234","stop",tok,"");
System.out.println("Response from WS:" + res);
```

Com es pot veure, es fixa l'adreça IP de destí per comprovar que el vídeo s'envia correctament, establint també els paràmetres necessaris per al mètode *createTXMediaServer*. S'han fet també proves amb el control dels vídeos per poder aturar-los o fer una pause (mètode c*ontrol*). La resposta correcta que s'ha de rebre és un 200 OK .

Es comprova doncs a l'equip receptor i amb un client VLC que el vídeo s'està rebent correctament i que al fer l'stop, el vídeo es para sense cap problema.

Pel que fa a l'Upload, es fa un client semblant per accedint directament als mètodes d'aquest servei.

```
String res;
String filename = "park_guell.m2t";

DataHandler dh = new DataHandler(new FileDataSource("D:/TFC/" + filename));
UploadServiceservice servicea = new UploadServiceserviceLocator();
System.out.println(servicea.getUploadServiceAddress());
UploadService porta = servicea.getUploadService();
System.out.println("Port:" + porta);

res = porta.publishResource(dh, filename ,tok,"");
System.out.println("Response from WS:" + res);
```

En aquest cas es puja un arxiu que està en local (D:/TFC) al path que s'ha especificat als scripts configurats abans. En aquest cas, el missatge si tot a sortit correctament també es un 200 OK. Pel cas que no trobés l'Api del VideoLAN tornaria el missatge: "No hay Apis activas".

Amb aquest client es pot comprovar que es puja l'arxiu especificat anant directament al path on s'emmagatzemen tots el vídeos.

4.1.2 Client per les proves del servei de Download

Per al cas del client del Download, s'ha considerat fer la mateixa prova que als altres dos serveis, fent una crida directa al mètode (amb una autenticació prèvia al servidor d'autenticació del projecte XAC, igual que en el cassos anteriors). El codi que correspon és el següent:

```
DownloadServiceservice serviced = new DownloadServiceserviceLocator();
System.out.println(serviced.getDownloadServiceAddress());
DownloadService_PortType portd = serviced.getDownloadService();
System.out.println("Port:" + portd);

Object ob = portd.DownloadResource("fires1.avi", tok, "");
if (ob instanceof javax.activation.DataHandler)

    DataHandler dh = (DataHandler)ob;
    BufferedOutputStream out = new BufferedOutputStream(new FileOutputStream("D:/Fires1.avi"));
    BufferedInputStream in = new BufferedInputStream(dh.getInputStream());

        byte[] buffer = new byte[256];
        while (true) {
           int bytesRead = in.read(buffer);
           if (bytesRead == -1)
              break;
           out.write(buffer, 0, bytesRead);
        }
        in.close();
        out.close();

        System.out.println("File Downloaded successfully");
```

Aquest client demana un vídeo (fires1.avi) al mètode *DownloadResource* i el va llegint per a guardar-lo al path D:/ amb nom Fires1.avi. Si aquest procés es correcte, es rep el missatge de "*File Download successfully".*

Es comprova també que es descarrega en local el vídeo seleccionat per verificar el seu funcionament.

4.2 Problemes durant el procés d'implementació i prova dels serveis

Com qualsevol aplicació que es vol crear o qualsevol servei que es vol donar, en tot el procés d'implementació i publicació sorgeixen diferents problemes que es van solucionant al llarg de la creació. En la implementació d'aquest WEB Service també s'han trobat alguns problemes tant per la part d'equips, temps o personal, com per la part d'implementació e instal·lació de les aplicacions o programes necessaris.

El primer problema que va sorgir va ser el fet d'obtenir un equip per a poder instal·lar aquests serveis. Com que inicialment no hi havia un equip dedicat a aquest projecte (XAC), es va duplicar el servei del PI a la mateixa màquina, però el fet de tindre dos Servidors WEB amb el mateix servidor Tomcat, donava problemes de convivència i com que es necessitaria un equip dedicat, es va demanar un. En aquest nou equip s'han implementat els tres serveis seguint els passos tal i com s'ha indicat als capítols anteriors.

Degut a la presentació del projecte els dies 17 i 18 de maig al MAC, es va haver d'implementar el WEB Service en un temps mes limitat, però no va donar temps a implementar totes les funcionalitats, sinó amb un funcionament mínim i amb tots els continguts dels usuaris en local per a fer les demostracions.

Una vegada es va desplegar correctament totes les funcionalitats dels serveis d'aquest mòdul, van arribar els problemes d'interacció amb els altres mòduls (fets per altres grups de recerca o universitats). Tot i haver creat els documents WSDL per cada part, hi van haver-hi certs problemes d'incompatibilitat que es van anar resolvent a mida que ens reuníem els diferents grups.

A més d'aquests problemes d'incompatibilitat, la falta d'estabilitat dels diferents mòduls implementats feia també difícil alguns dies, el fer proves d'aquest mòdul ja que, per exemple, si el WEB Service d'autenticació no funcionava, no es podia accedir al nostre servei.

Aquest últim problema, es resoldrà amb la integració de tots els mòduls als dos servidors que s'han comprat pròpiament pel projecte XAC, i que es quedaran estables al MediaCAT amb una certa gestió i control.

Una vegada solucionat els problemes d'interacció entre els mòduls, i amb els serveis d'aquest WEB Service, funcionant correctament, es va veure que a l'hora d'autenticar-se es feia servir el mòdul d'autenticació del PI, per tant, el grup que implementava la part WEB, al voler fer la interacció amb els serveis de Upload/Download i Streaming, van localitzar l'error i per tant s'havien de canviar les classes per les noves, fet que va provocar que aquest serveis deixessin de funcionar per un problema amb el servidor Axis.

Per solucionar aquest problema es va demanar al grup amb el servei d'autenticació que revisés el nostre servei per localitzar el error. Finalment es va tornar a muntar des de zero tots tres serveis i es va aconseguir que funcionés correctament.

En la següent demostració amb les televisions locals, va sorgir un problema amb un dels mètodes que controlava el flux de vídeo i no es podia fer un *stop* de l'Streaming d'un vídeo directament des de l'aplicació ni de la pàgina WEB, per tant es va haver de fer manualment parant el flux quan es feia clic a l'stop. Aquest problema era degut a que al modificar l'arxiu principal de l'Streaming es va eliminar una comanda que guardava les sessions de cada un dels Streamings, llavors no es guardava la sessió i per tant quan es cridava, no trobava cap sessió per parar. Es va revisar tot el codi i va solucionar el problema.

Un altre problema que es va tindre va ser a l'hora de provar els vídeos de les televisions, ja que el format que inicialment havien utilitzat per donar-los vídeos d'exemple estaven en format *DV RAW* i no permetia fer l'Streaming per la previsualització de cadascun dels vídeos, però això va tindre fàcil solució, ja que aquest format no era el definitiu (a l'estudi inicial dels formats que s'utilitzaven en cada cas es va fixar l'MPEG1 per a l'Streaming, capítol 1.2.4), per tant es van transcodificar tots els vídeos a MPEG1 i es va solucionar el problema.

Finalment, tot i tindre problemes com els que s'acaben de comentar (i d'altres amb menor importància), els serveis van quedar "estables" i funcionant tal i com s'esperava.

5 Capítol 5. Costos

Qualsevol projecte que es vol dissenyar e implementar va lligat a una sèrie de gestions que s'han de tenir en compte. Una de les gestions mes importants del projecte i que s'ha de considerar al principi de l'estudi, són els costos tant materials com de personal.

En aquest capítol es farà l'estudi dels costos dels dos tipus, associats tant a la implementació realitzada, com al disseny de la implementació amb una certa gestió i estabilitat estudiada al capítol 2.5.

5.1 Costos de la implementació realitzada

En aquest apartat, es veurà de forma ràpida, els costos tant econòmics com de temps, que s'ha necessitat per a dur a terme aquesta part del projecte XAC:

5.1.1 Costos econòmics

A la següent taula es poden veure els costos que han suposat el fet d'implementar aquesta part del projecte XAC, on s'ha tingut en compte els costos de material i de personal:

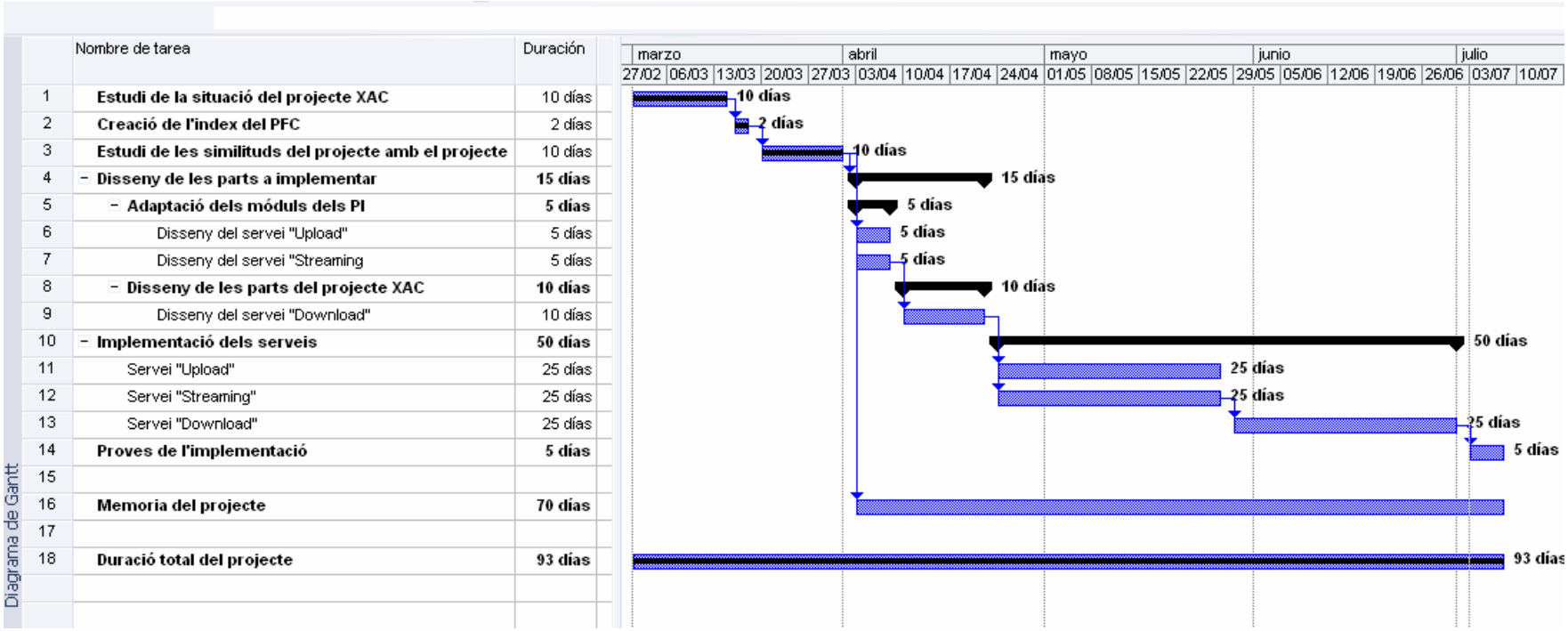
Costos materials	Equip servidor 1.500 €
Costos de personal	Enginyer (4h/dia, 7€/h, 4 mesos) 2.240 €
Costos de manteniment (a partir de la finalització del projecte)	Enginyer 900 €/mes
TOTAL	**3.740 €**

Taula 1. Costos en equips i personal del projecte

El cost total del projecte és de 3.740 €, tal i com indica la Taula 1, considerant que després de la finalització, farà falta continuar amb una persona que dediqués temps a fer-li el manteniment, actualitzacions o millores que considerés oportunes, per tant, s'hauria de sumar al cost total, el sou d'aquest enginyer.

5.1.2 Costos de temps

Per la part de temps invertit, el cost hores dedicades al disseny i la implementació d'aquest ha estat 4h diàries (dies lectius) durant 4 mesos i que s'han repartit de la següent manera:



Il·lustració 16. Tasques realitzades al projecte

5.2 Costos del disseny estable

Pel cas dels costos per al disseny amb una certa estabilitat s'ha de tindre en compte per exemple que el temps de realització serà més gran en cas que només hi hagi un enginyer, per tant es considera necessari un segon enginyer per duplicar el servei, i per tant els costos seran més elevats. A més, al voler fer un disseny en que es dupliquen els servidors, s'ha de tindre en compte també que s'hauran de comprar més equips, i per tant també augmentarà el cost per aquesta banda. A continuació es veu amb més detall aquests costos.

5.2.1 Costos econòmics

A la següent taula es mostra quins serien els costos econòmics en cas que es volgués muntar l'estructura explicada a l'apartat 2.5, amb una certa estabilitat i escalabilitat. Inicialment es considera que es muntarien dos clústers d'equips, llavors per aquest projecte s'ha de considerar de comprar un altre equip més:

Costos materials	2 Equip servidor 1.500 € x2 3.000€
	1 Equip servidor VideoLAN 1.500 €
Costos de personal	2 Enginyer (4h/dia, 7€/h, 4 mesos) 2.240 € x2 4.480€
Costos de manteniment (a partir de la finalització del projecte)	2 Enginyer 900€/mes x 2 1.800 €
TOTAL	**8.980 €**

Taula 2. Costos en equips i personal a llarg termini

Es considera que, inicialment, es muntarien dos clústers de servidors a dos llocs diferents, per tant farien falta dos enginyer a dos llocs diferents, per donar més estabilitat, i tant per tota la configuració dels serveis com pel manteniment dels servidors.

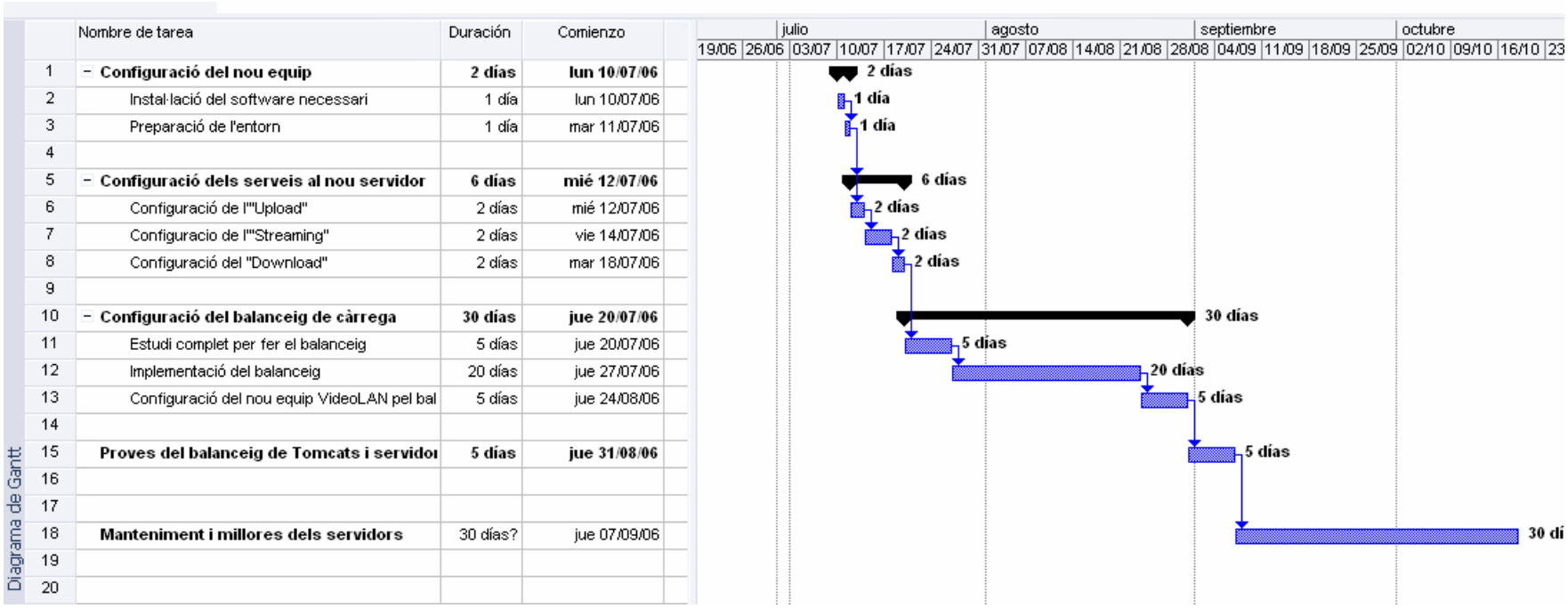
Es compraria també un nou equip servidor que contindria l'API de VideoLAN per poder balancejar també la càrrega dels vídeos, no només les del Tomcat.

Els costos de manteniment, com s'indica a la Taula 2, s'haurien de considerar només per després de tota la implementació, per això no estan dintre del cost total, com en el cas dels costos de la implementació actual, Taula 1.

5.2.2 Costos de temps

En aquest cas, la part de la implementació dels serveis ja estaria feta tal i com indica la Il·lustració 16, per tant, només caldria configurar correctament el segon servidor, configurar correctament el balanceig de càrrega i fer la gestió

de tot el clúster. A continuació es mostra com quedaria aquesta repartició de tasques, considerant el temps a partir de la finalització de la implementació anterior i amb unes dates aproximades tant de duració com de repartició de la feina:



	Nombre de tarea	Duración	Comienzo
1	**Configuració del nou equip**	**2 días**	**lun 10/07/06**
2	Instal·lació del software necessari	1 día	lun 10/07/06
3	Preparació de l'entorn	1 día	mar 11/07/06
4			
5	**Configuració dels serveis al nou servidor**	**6 días**	**mié 12/07/06**
6	Configuració de l'"Upload"	2 días	mié 12/07/06
7	Configuracio de l'"Streaming"	2 días	vie 14/07/06
8	Configuració del "Download"	2 días	mar 18/07/06
9			
10	**Configuració del balanceig de càrrega**	**30 días**	**jue 20/07/06**
11	Estudi complet per fer el balanceig	5 días	jue 20/07/06
12	Implementació del balanceig	20 días	jue 27/07/06
13	Configuració del nou equip VideoLAN pel bal	5 días	jue 24/08/06
14			
15	**Proves del balanceig de Tomcats i servidor**	**5 días**	**jue 31/08/06**
16			
17			
18	**Manteniment i millores dels servidors**	30 días?	jue 07/09/06
19			
20			

Il·lustració 17. Tasques a realitzar una vegada finalitzat aquest projecte

6 Conclusions

Finalment, i per concloure amb aquest projecte, es presentaran unes conclusions generals del mateix:

- S'ha fet un estudi inicial de les possibilitats per fer la implementació dels serveis i s'ha escollit finalment el mateix mètode que va utilitzar el PI: xarxes peer-to-peer amb JXTA.
- S'han estudiat parts del PI amb la finalitat d'aprofitar les parts ja implementades per aquest projecte ja que te característiques semblants amb el projecte XAC.
- S'han adaptat aquestes parts a les especificacions inicials del projecte XAC i s'han implementat les parts que no estaven al PI.
- S'ha realitzat un estudi previ de la possibilitat de fer aquest mateix projecte amb una certa escalabilitat que serà necessària en un termini curt de temps.
- S'han realitzat les proves oportunes de cara a la integració amb les altres parts del projecte.
- S'ha configurat correctament el nou servidor VideoLAN al mateix equip comprat per la XAC (tot i que encara no està en marxa).

S'han complert per tant els objectius inicials d'aquest projecte, per tant, ara, a curt termini, les tasques principals que quedaran per fer seran:

- La implementació del balanceig de carrega per acabar de donar una certa estabilitat al servei. Aquest és el tema més important, ja que actualment si utilitzem únicament un servidor VideoLAN el numero de fluxos que es poden servir a la vegada, com s'ha vist a l'apartat 4.1, és de 2-3. Amb la duplicació del servidor, podríem anar doblant aquest numero.
- Configurar els clústers per al balanceig de Tomcats (per les peticions dels altres serveis), ja que el balanceig de VideoLAN només ens permet fer el balanceig en el cas del servei d'Streaming, per als altres serveis, ens faran falta més equips configurats amb Tomcat.
- Manteniment i control dels serveis: controlar (amb ajuda de la gestió feta amb Nagios) que els serveis estiguin sempre en marxa per donar una bona estabilitat dels serveis.

Amb aquestes últimes consideracions, aconseguirem que el nostre WEB Service tingui una certa estabilitat en quants als serveis que es donen i a més escalabilitat davant de les peticions d'usuaris a la vegada.

6.1 Impacte mediambiental

El tema de l'impacte mediambiental és molt obert en molts aspectes, ja que hi han moltes coses que es poden considerar dintre d'aquesta branca.

Si es considera només el fet de que s'ha realitzat un projecte de software, llavors es pot dir que no causaria cap impacte mediambiental, ja que s'ha utilitzat únicament la programació d'aplicacions i, en principi tot es faria dintre d'un laboratori habilitat per aquest tipus de feina, però si mirem més enllà d'aquest projecte, i fem una visió global del projecte XAC sencer, tornant a la filosofia inicial que volia donar el projecte, "facilitar l'intercanvi (compravenda) de continguts entre les televisions locals", llavors podem considerar altres aspectes com per exemple:

- Com que les televisions utilitzaran els seus equips per canviar els seus continguts, no necessitaran de missatgers o del fet de desplaçar-se amb els seus vehicles, per tant es pot considerar que no faran tant d'ús dels vehicles, i llavors menys contaminació.

Per altra banda, si es considera que darrera del projecte de software hi han equips (consum d'aigua i productes químics per fabricar-lo, electricitat per mantenir-lo...) llavors l'aspecte positiu d'aquest tipus de projectes desapareix.

Seguint un estudi fet pel XTEC[9] (Xarxa telemàtica d'educació de Catalunya) i que es pot trobar a l'enllaç [11], es poden treure conclusions sobre el problema mediambiental que te el fet d'utilitzar equips informàtics. A continuació he tret alguns exemples que l'estudi dóna:

- Per produir un chip de memòria (32Mbybes DRAM) de 2 grams, s'utilitzen 1600 grams de combustible fòssil, 72 grams de químics i 32 litres d'aigua (veure [12])
- Per produir un PC d'escriptori amb el seu corresponen monitor CRT, s'utilitzen 290 kg de combustible fòssil, 22 kg de químic i 1500 litres d'aigua (veure [13])
- De tota l'electricitat que consumeix un ordinador al llarg de la seva vida (considerant només tres anys d'ús), el 83% es va utilitzar en el procés de producció i el 17% restant és l'electricitat que consumeix en el seu ús diari. (veure [13])
- El consum d'electricitat d'una planta fabricant de chips representa al voltant del 40% dels costos de producció, sobretot degut als ventiladors i bombes d'aire (veure [14])
- Una planta fabricant de chips consumeix 7 milions de litres d'aigua cada dia (veure [14])

Amb aquestes dades ja podem extraure conclusions sobre l'impacte que te el fet d'utilitzar PCs servidors en aquest projecte. A més a més, i seguint amb el mateix estudi, podem veure el problema diari que te el fet d'utilitzar-los:

- A l'any 2000, en EEUU els equips d'oficina i telecomunicacions ja consumien el 3% de l'electricitat nacional. Els ordinadors representaven el 43% d'aquest consum.

[9] Pàgina principal: www.xtec.es

- Un PC d'escriptori (Pentium IV) amb el seu monitor (17 polzades) usa aproximadament uns 118W de potencia en mode actiu (veure [12])
- Es calcula que tan sol el 25% dels ordinadors tenen correctament configurat el mode de baix consum.
- Una pantalla de cristall líquid consumeix entre un 60 i 70 % menys que una pantalla de raig catòdics. Els portàtils són també més eficients energèticament que els PC d'escriptori convencionals.

A més d'aquests fets del dia a dia i que repercuteixen directament en el consum de recursos a nivell mundial, tenim els fets d'haver de desfer-nos de l'equip, una vegada acabat el temps en que l'equip es "eficient". Aquest tema també és molt important, ja que segons aquest mateix estudi, per exemple:

- El 90% dels equips informàtics antics acaben als abocadors, després d'haver estat llançats al contenidor, abandonats al carrer o dipositats a les ferralles.
- Es va calcular que al 2005, els PCs obsolets a EEUU ocupaven 5.7 milions de m^3 (l'equivalent a un camp de futbol de 1.5 km d'altura)
- Al 2005 les escombraries electròniques ja representen el 5% de tots els residus generats per la Unió Europea. A Espanya generem cada any 200.000 tones d'escombreries electròniques. Només reciclar els ordinadors que avui dia s'amuntonen als abocadors europeus portaria uns 10 anys.

Amb aquestes dades (poc considerades moltes vegades) es pot veure, que estudiant a fons l'impacte en projecte de software també es pot treure conclusions negatives, tot i així, aquests punts negatius es van estudiant dia darrera dia i es va solucionant amb mètodes per exemple de reciclatge o reutilització.

7 Bibliografia

[1] Pàgina principal de la *Fundació i2CAT*: www.i2cat.net (Data de consulta: 27/03/2006)

[2] *Aplicacions P2P* : http://apuntes.rincondelvago.com/aplicaciones-p2p.html (Data de consulta: 17/03/2006)

[3] *Xarxes P2P*: www.infor.uva.es/~jvegas/docencia/aso/p2pjxta2.0.ppt (Data de consulta: 17/03/2006)

[4] *Redes P2P i JXTA*: http://www.inf.utfsm.cl/~rmonge/sd/trabajos/trejo-zu%F1iga-p2p.pdf (Data de consulta: 18/03/2006)

[5] Clúster de computadors: http://es.wikipedia.org/wiki/Cluster_de_computadores (Data de consulta: 3/05/2006)

[6] *Axis installation instructions* : http://ws.apache.org/axis/java/install.html (Data de consulta: 15/04/2006)

[7] *Axis: the next generation of Apache Soap* http://www.javaworld.com/javaworld/jw-01-2002/jw-0125-axis.html (Data de consulta: 15/04/2006)

[8] *NRPE* per Nagios: http://nagios.linuxbaja.org/?q=node/43 (Data de consulta: 20/06/2006)

[9] *Apache 2.x + Tomcat 4.x + Load Balancing (or Private JVMs):* http://raibledesigns.com/tomcat/ (Data de consulta: 25/06/2006)

[10] *Load Balancing at the Web Level with mod_jk*: http://www.redhat.com/docs/manuals/rhaps/jonas-guide/s1-load-balancing.html (Data de consulta: 25/06/2006)

[11] *Contaminación y material informático:* http://www.xtec.es/~acastan/textos/Contaminacion%20y%20material%20informatico.pdf (Data de consulta: 03/07/2006)

[12] "Environmental impacts of microchip manufacture", Eric D. Williams, http://dx.doi.org/10.1016/j.tsf.2004.02.049 (Data de consulta: 03/07/2006)

[13] "Revisiting energy used to manufacture a desktop computer: hybrid analysis combining process and economic input-output methods", Eric D. Williams, http://ieeexplore.ieee.org/iel5/9100/28876/01299692.pdf?arnumber=1299692. (Data de consulta: 03/07/2006)

[14] [OPC] "Revista Opcions nº 6: Els Ordinadors (páginas 8 a 12)", Opcions, http://cric.pangea.org/pdf/op62.pdf. (Data de consulta: 03/07/2006)

8 Annexes

8.1 Annex A: Manual d'utilització de l'e-Ruc

El primer que cal fer es descarregar l'aplicació des de la WEB oficial de la XAC. A data de 22 de juny del 2006, l'e-Ruc en la seva versió XAC 1.5 es troba disponible des de la URL de la facultat d'informàtica de Barcelona: http://fibi2cat2.fib.upc.edu/xac/zips/eRuc_XAC_v1.5.zip . Es tracta d'un fitxer de 13Mbytes.

Com encara no està en format Windows executable, cal descomprimir-lo manualment en qualsevol carpeta del sistema operatiu i executar el fitxer erucwin.bat.

El programa e-Ruc ve a ser una aplicació multiplataforma o màquina virtual basada en el sistema Java de la casa Sun Microsystems, que es pot executar en qualsevol tipus d'arquitectura (PC, Apple, Linux, telefons mòbils...) ja que no depèn dels dispositius que s'utilitzin. Com Java és un llenguatge de programació, cal tenir al menys la versió 1.5 de Java Runtime Environment instal·lada al nostre sistema, que és amb la que s'ha desenvolupat l'e-Ruc. Es pot baixar la última versió del software Java també des de la següent URL: http://www.java.com/es/Download/index.jsp. Per comprovar la nostra versió de Java sota l'entorn Windows XP es pot saber fent: Inicio -> Ejecutar -> "javaws". En cas de que tinguem alguna de les versions del software instal·lada en el sistema, ens sortirà l'aplicació següent (figura 1). En cas contrari, cal descarregar-lo des de la URL donada.

L'avantatge de Java per realitzar Streaming de vídeo, radica en que evita la necessitat d'un plugins addicionals per tal de poder reproduir-los, donat que la funció de reproducció ja es troba instal·lada en el sistema.

Il·lustració 18. Aplicació JAVA amb la última versió de Java (v1.5)

Un cop tenim instal·lat l'e-Ruc i abans d'executar-lo per primer cop, és recomanable disposar dels reproductor de medis multimèdia basats en software lliure següents:

- VLC (Video LAN Client) d'alta portabilitat per varis formats i códecs audiovisuals. A més es capaç d'assumir diferents rols dins de les tecnologies de Streaming ja que pot actuar tant com client reproduint el fitxer en temps real mentre es descarrega, com actuar de servidor per fer stream en monotasca o multitasca per protocols IPv4 o IPv6 de banda ampla. Es troba disponible a la URL:

    - http://fibi2cat2.fib.upc.es/eruc/zips/vlc.zip

- MPLAYER (The Movie Player). Probablement el reproductor basat en SW lliure més complert que existeix. Inicialment pensat per sistemes Linux, actualment funciona sota la majoria de plataformes. Igual que el VLC es troba disponible a la URL :

    - http://fibi2cat2.fib.upc.es/eruc/zips/mplayer.zip

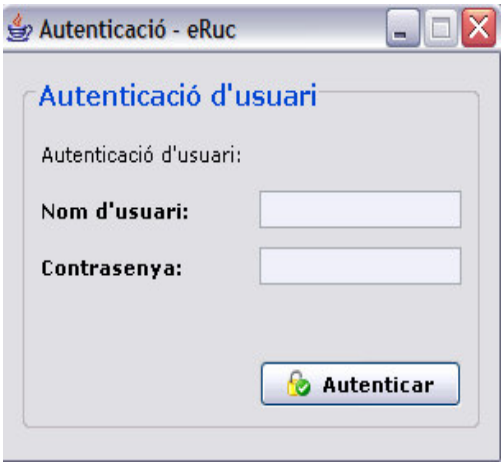
En versions posteriors de l'e-Ruc versió XAC està previst que l'instal·lador els integri.

8.1.1 Instal·lació i interfície gràfica

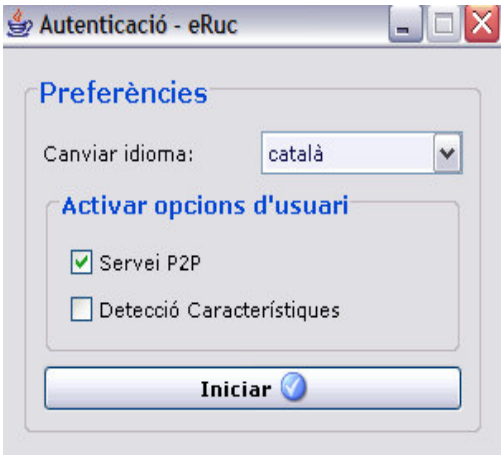
Si tenim plataforma Windows, per executar l'aplicació cal fer clic sobre el fitxer "erucwin.bat".

- El primer que ens trobem és l'autenticació en el sistema, demanant-nos un login i un password [figura 1]. En la fase de proves, tant una com l'altre són "admin", tot i que quan estiguin donats d'alta tots els usuaris tindran cadascun un de particular.

- Un cop estem validats al sistema accedim a un primer pop-up de preferències [figura 2] que ens dóna a escollir canviar l'idioma triant entre castellà, català o anglès i quin tipus de servei volem activar. Per defecte ens ve activat únicament el check box de servei P2P, per tal de poder connectar directament validant-nos i autenticant-nos dins de la xarxa. També és possible activar "Detecció Característiques" si el client és un dispositiu mòbil i volem que s'apliqui la millor configuració possible a l'hora de treballar amb els continguts audiovisuals, com per exemple códecs, resolucions...etc. Cal anotar però, que aquesta opció estarà en principi desactivada per la versió definitiva de l'e-Ruc adaptada a la XAC. Per últim només cal validar la nostre selecció fent clic al botó d' iniciar.

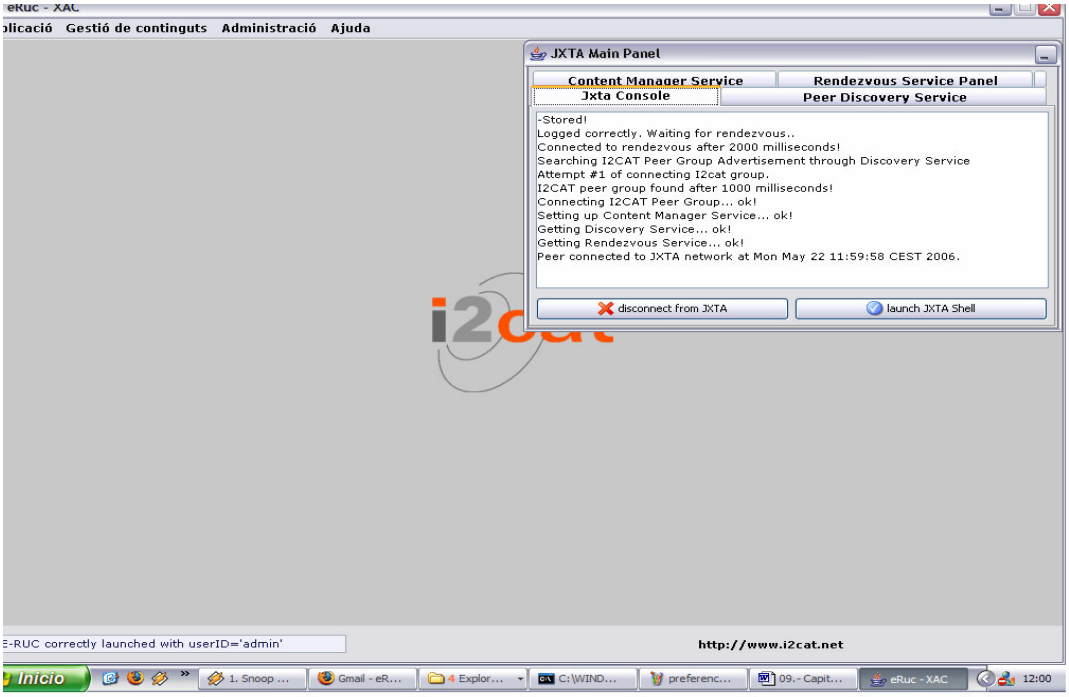


Il·lustració 19 Autenticació dins la xarxa



Il·lustració 20 Preferències

Un cop validats, ens apareix la consola principal de l'aplicació amb configuració de servei P2P [figura 4].



Il·lustració 21 Consola principal de l'e-Ruc

La consola es composa d'una interfície força austera. Apart de la clàssica barra de menús, apareix una finestra "JXTA Main Panel" que engloba mitjançant l'ús de diferents solapes tot l'estat de la connexió dins de la xarxa i ens mostra en tot moment l'estat de l'usuari dins d'ella. Probablement en versions posteriors del programa, algunes d'elles seran transparents de cara a l'usuari ja que no cal una supervisió tant exhaustiva.

En consonància amb el programa que està implementat amb Java, les finestres estan programades en format Xform. Xform és un tipus de formulari basat en tecnologia XML més dinàmic i que ofereix més funcionalitats que els actuals Forms. Les tecnologies XML són un conjunt de mòduls que serveixen per estructurar, emmagatzemar e intercanviar informació mitjançant diferents aplicacions. En efecte, hom pot considerar que XML és un llenguatge o format d'etiqueta extensible molt simple i similar al HTML però la seva funció principal és descriure dades i no mostrar-les com és el cas de HTML.

8.1.2 Barra de menús

Com qualsevol programa al desplegar la barra de menús es troben totes les funcionalitats possibles de l'aplicació.

8.1.2.1 APLICACIÓ

Dins del primer menú contextual, trobem quatre opcions principals:

- Opcions: A la primera revisió de l'e-Ruc, només se'ns dóna la possibilitat de poder canviar l'idioma. Podem triar entre el castellà (per defecte), el català i l'anglès.

- Iniciar JXTA: És el cor de l'aplicació en quant a la connectivitat es refereix. Es carrega de forma automàtica quan arranquem el programa, per tal de donar-nos accés a la xarxa mitjançant el protocol JXTA. Tant si s'inicia de forma manual com de forma automàtica, l'estat de la connexió es pot fer remota des de la finestra (JXTA Main Panel) encarregada del control de l'estat dins de la xarxa.

- Detecció de característiques: Es desplega una nova finestra on ens realitza una detecció de la nostra configuració si som un dispositiu de telefonia mòbil.

- Sortir: Desconnecta i surt de l'aplicació e-Ruc sense demanar cap tipus de confirmació.

8.1.2.2 GESTIÓ DE CONTINGUTS

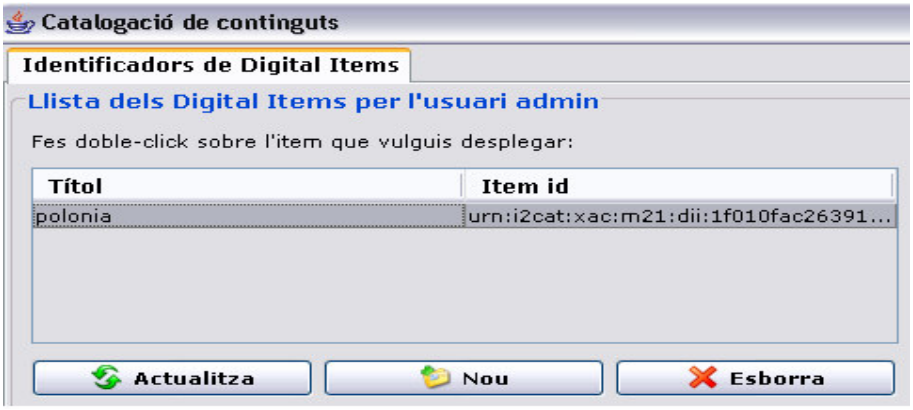
És el mòdul central de tota l'aplicació, ja que contextualitzant, serà l'encarregada de fer la compra i/o venda de continguts entre els diferents usuaris de la xarxa.

Per tal d'explicar el seu funcionament podem separar-la en dues vessants diferents: catalogació (que serà la venda de continguts) i cerca (que farà la crida a la compra d'un contingut) dels diferents digital items (DI d'ara en endavant).

8.1.2.3 CATALOGACIÓ DE CONTINGUTS

La primera de les dues opcions anteriors fa esment a la situació de que un usuari vulgui fer una catalogació de continguts, és a dir, es necessita fer una venta d'un DI del que es disposin els drets i per tant posar-lo en disposició dels demés usuaris de la XAC en base a una llicència.

En primera instància apareix el dialog associat [veure figura 4] el qual ens mostra una taula on surten enllistats els DI prèviament catalogats que tenim disponibles de forma local, ja siguin continguts propis o descarregats prèviament de la xarxa, ordenats per títol amb la seva informació associada (item id).



Il·lustració 22. Finestra de catalogació de continguts amb un fitxer prèviament catalogat

L'aplicació ens permet fer a partir d'aquí principalment 3 funcions, fer un refresc dels DI presents al sistema amb "actualitza", esborrar algun DI amb "esborra" o començar pròpiament la catalogació a partir de fer clic al botó "Nou". Cal escollir, un cop s'ha fet clic, entre dues opcions:

- Crear la catalogació a partir d'un nou DI no enllistat que l'usuari vulgui compartir o posar disponible cap a la xarxa.

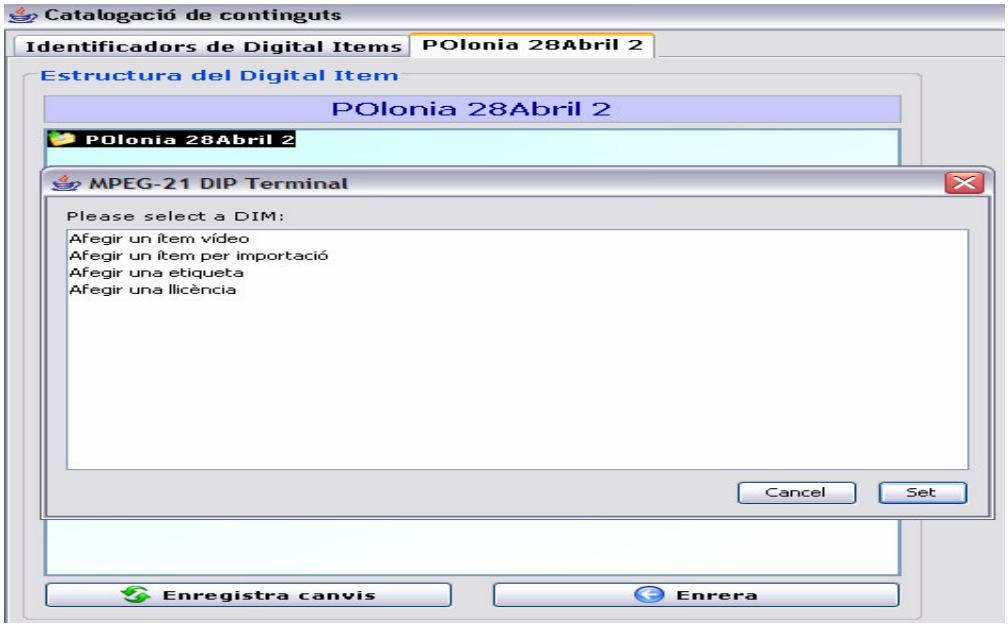
- Fer una "recatalogació" de DI disponibles.

Tant en el cas de voler catalogar un nou DI, com si volem recatalogar un d'existent cal tenir en compte que haurà d'estar donat d'alta en tres mòduls diferents: continguts, metadades i llicències.

En primer lloc cal especificar quin tipus de contingut i quin fitxer físic incorporarem o volem modificar del nostre sistema o de la xarxa. Per tal de simplificar l'explicació, en els propers punts s'explicarà com tractar de publicar un nou recurs o projecte amb els seus DI associats. Modificar-ne un d'existent, serà, paral·lelament una variació d'aquest.

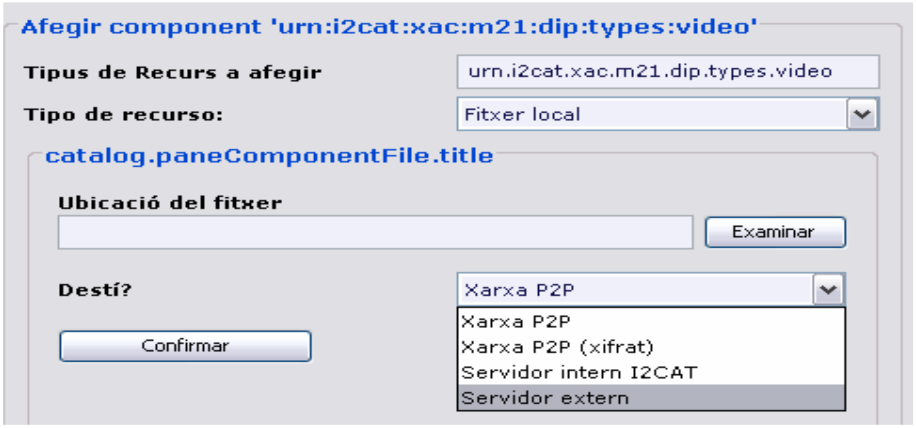
Per crear un recurs nou, cal "Crear un digital item nou", s'escull un títol representatiu del contingut i es tria el tipus de DI del que es vol catalogar i validem acceptant. Una forma d'optimitzar la feina seria crear una plantilla per a cada tipus d'esdeveniment multimèdia (àudio, vídeo...etc) per tal d'agilitar la feina de catalogació en múltiples DI.

Com es veu a la figura 5, apareix una nova solapa amb nom del recurs obert anteriorment, on al fer clic a sobre de la icona amb el botó secundari del mouse, ens apareix una nova finestra amb totes les accions que podem dur a terme entre elles "afegir un item de vídeo" nou o aprofitar un d'existent "Afegir un item d'importació" dintre del projecte en curs (Polònia 28Abril2 en l'exemple). És a dir, dintre d'un DI podem afegir components discrets diversos. En el cas de que es vulgui canviar el nom del projecte cal "Afegir una etiqueta". El camp "afegir una llicència" serà comentat en punts posteriors.



Il·lustració 23. Finestra Xform de catalogació de continguts

Per últim, cal seleccionar el tipus de recurs, així com la seva ubicació, d'aquesta forma podem triar entre en local (en el cas de que tinguem el DI en el nostre sistema) o bé des de Internet via una URL externa (si es troba en una altra màquina fora de la nostra xarxa local, com per exemple un servidor de vídeo extern, on caldria determinar de quin tipus de servidor es tracta (FTP,HTTP,MMS) i quina és la seva adreça. També es dona l'opció de si volem aprofitar per transcodificar el clip de vídeo de forma automàtica contra un altre tipus de format dels disponible a la xarxa (transcodificat). Fent ús d'aquesta característica, evidentment es cridarà de forma solidària al mòdul de transcodificació.



Il·lustració 24. Catalogació d'un nou DI a nivell de contingut

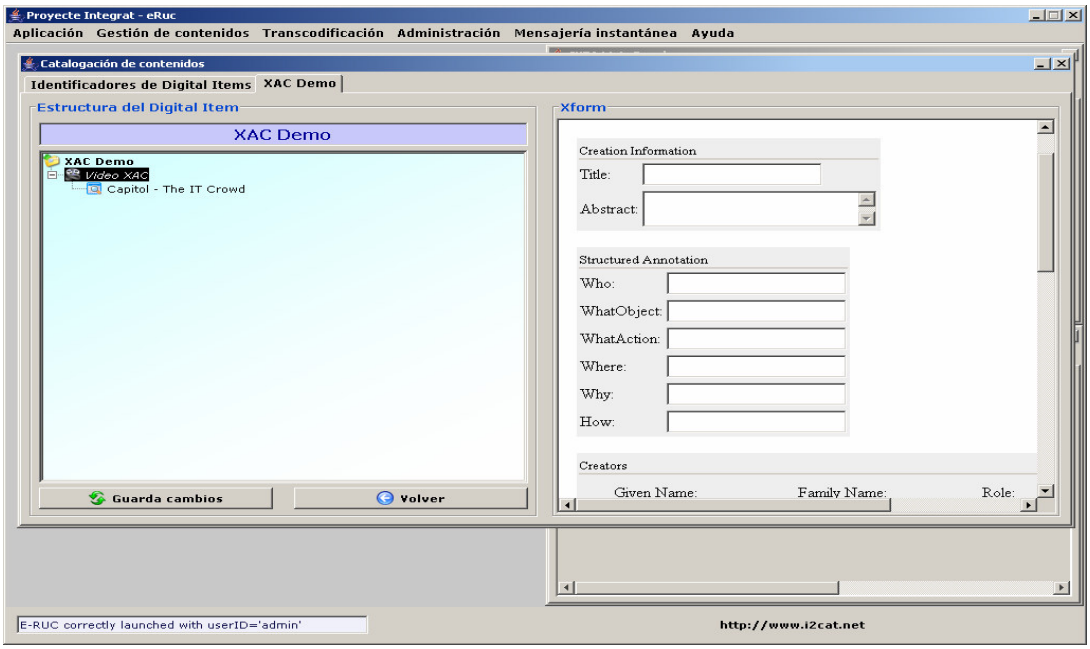
En el cas de que es disposi del fitxer en local, caldrà decidir on volem publicar el contingut [figura 6]. Hi ha tres destins possibles bàsicament:

- Xarxa P2P, on quedaria guardat en el nostre sistema tot i que si per qualsevol raó la computadora no estigués operativa, el recurs no estaria disponible vers la xarxa. La opció de "xifrat" seria el mateix cas únicament encriptant el recurs.
- Servidor extern defineix que el DI romandrà en un servidor extern propi de l'usuari, estigui o no dins de les seves instal·lacions, i per tant hi serà sempre disponible. Aniria en consonància amb la mateixa filosofia P2P "domèstica".
- Servidor intern i2CAT, en aquest cas es defineix que es farà un Upload contra un servidor de continguts aliè a l'usuari.

Un cop tot especificat només cal confirmar la nostra selecció. Notar que en tot moment podem tornar a la pantalla anterior amb el botó creat a tal efecte.

La catalogació respecte a les metadades és substancialment diferent. Únicament cal omplir les especificacions dels camps del Xform vinculat a les metadades anomenades descripcions a l'e-Ruc. Un cop validades es publiquen les metadades establertes com a públiques al servidor de metadades i drets.

Aquests camps venen donats per un consens en la fase prèvia d'anàlisi del projecte, entre les diferents seus audiovisuals adherides al projecte XAC, sota la normativa del format MPEG-21 [figura 7 tot i no ser la interfície definitiva]. Com aquestes metadades estan dissenyades en llenguatge XML es defineixen etiquetes en funció del tipus de dades que siguin. Notar que en tot moment, hom pot afegir, editar o esborrar metadades.



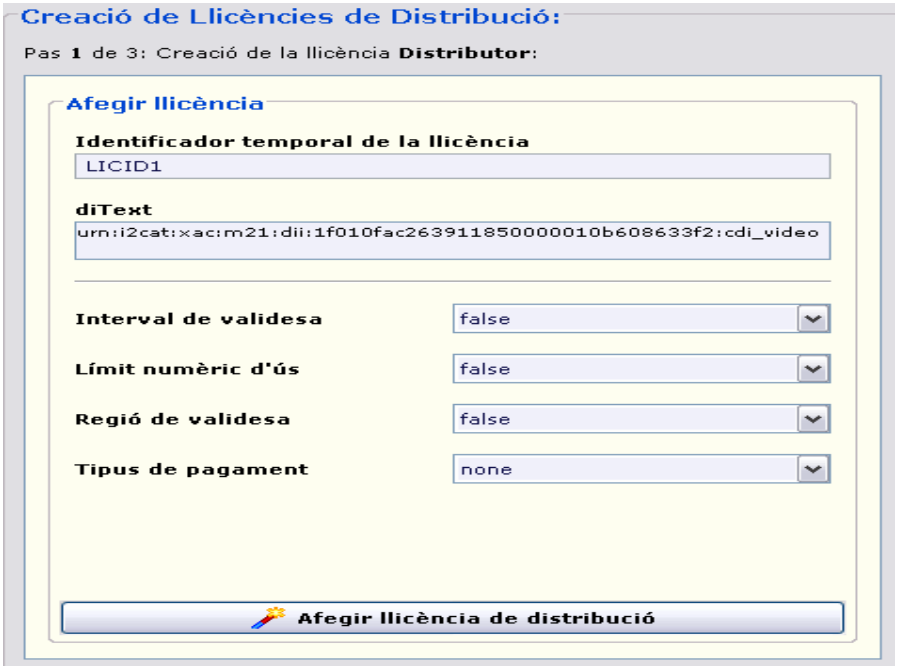
Il·lustració 25. Catalogació d'un nou DI a nivell de metadades

L'últim pas per tal de catalogar un DI un cop el tenim identificat i catalogat es deixar-ho a disposició dels demés usuaris per tal de que el puguin consumir.

Per tal de fer això cal, però, assignar al mateix DI uns drets, una llicència i uns paràmetres i condicions de vigència d'ús vers al client final, que ens validarà la vida i condicions d'aquest DI en el sistema final de l'usuari. És a dir, si el podran modificar, només reproduir, si s'haurà d'esborrar un cop consumit el recurs...etc.

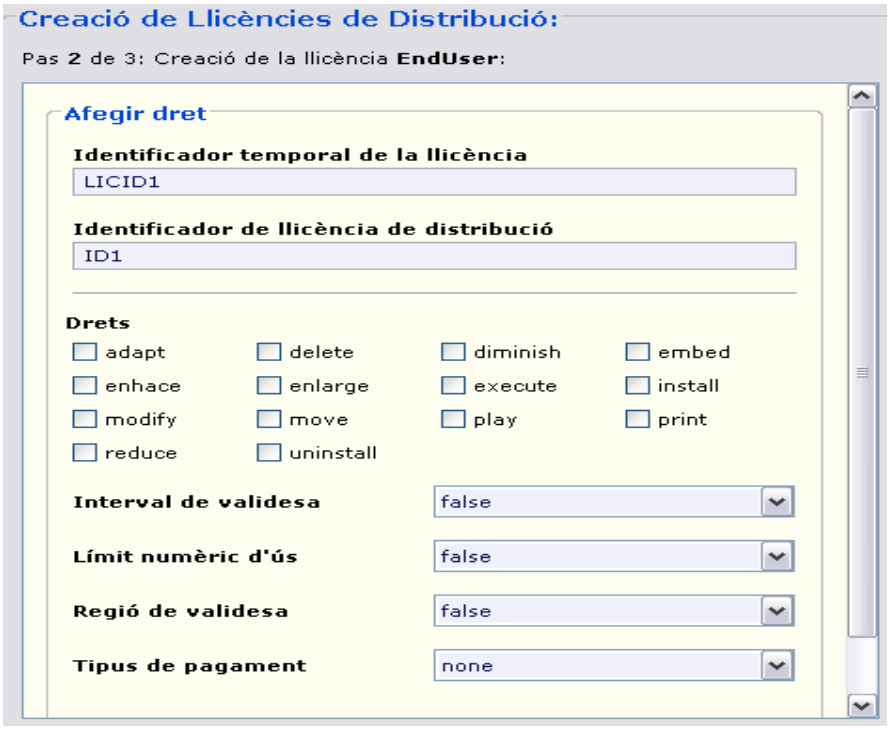
Per tal d'establir la llicència cal partir de "Afegir una llicència" de la figura 5. La tasca de creació de les diverses llicències de distribució es divideix en tres passes diferents.

La primera és la creació pròpiament de les característiques de la llicència, assignant mitjançant la selecció dels quatres combo box, les principals característiques de la mateixa i validant l'acció amb "Afegir llicència de distribució" [figura 8].



Il·lustració 26. Creació d'una llicència de distribució. Pas 1

El segon pas és establir quins seran els privilegis de cara a l'usuari final consumidor del DI publicat així com el tipus de pagament a realitzar [figura 9]. Cal notar que encara no està amb la seva interfície definitiva a data 30 de maig 2006.



Il·lustració 27. Establiment de llicències i drets per l'usuari final. Pas 2

L'últim pas és una confirmació de l'establiment de la llicència en els termes i drets estipulats, tant per la part del venedor (catalogador) com del comprador (usuari final). Un cop queda validada ens apareix un nou Xform confirmant l'alta de la llicència i ens retorna un identificador [figura 10].

Il·lustració 28. Confirmació de l'establiment de la llicència. Pas 3

Per una altra banda, recordar també que es possible en qualsevol moment actuar sobre el fitxer i modificar qualsevol dels tres mòduls de catalogació, ja sigui a nivell de continguts, metadades o drets / llicències.

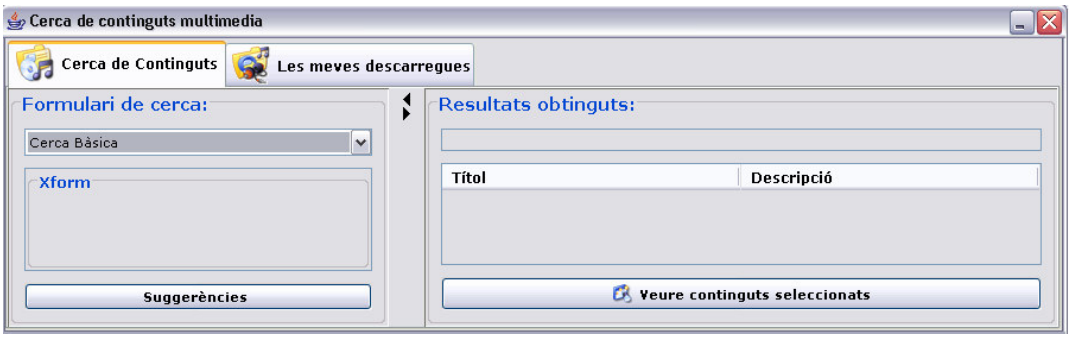
Efectivament, al desplegar el dialog associat al recurs que volem utilitzar, veiem les accions que podem realitzar, on en principi, totes les accions estan estandarditzades per a cada tipus de DI. D'aquesta manera, podem seleccionar diferents esdeveniments estipulats per aplicar: com reproduir, esborrar, actualitzar informació d'última hora, afegir o eliminar qualsevol tipus de dret o llicència, editar algun camp de les metadades...

Cal apuntar que en qualsevol moment podem gravar els canvis o tirar enrere desfent els canvis no guardats fins al moment de fer clic al botó pertinent.

8.1.2.4 CERCA DE CONTINGUTS

La segona de les dues opcions sobre la gestió de continguts es dóna precisament quan l'usuari vol realitzar una consulta sobre algun contingut que pugui ser del seu interès i si s'escau en últim terme, un cop s'hagi consultat la previsualització del contingut (encara no implementada en la versió 1.2 de l'e-Ruc versió XAC) i quins drets i tipus de llicències té, realitzar la compra del DI.

Si des de la barra de menús, seleccionem Cerca de Continguts ens apareix un nou Dialog [figura 10].

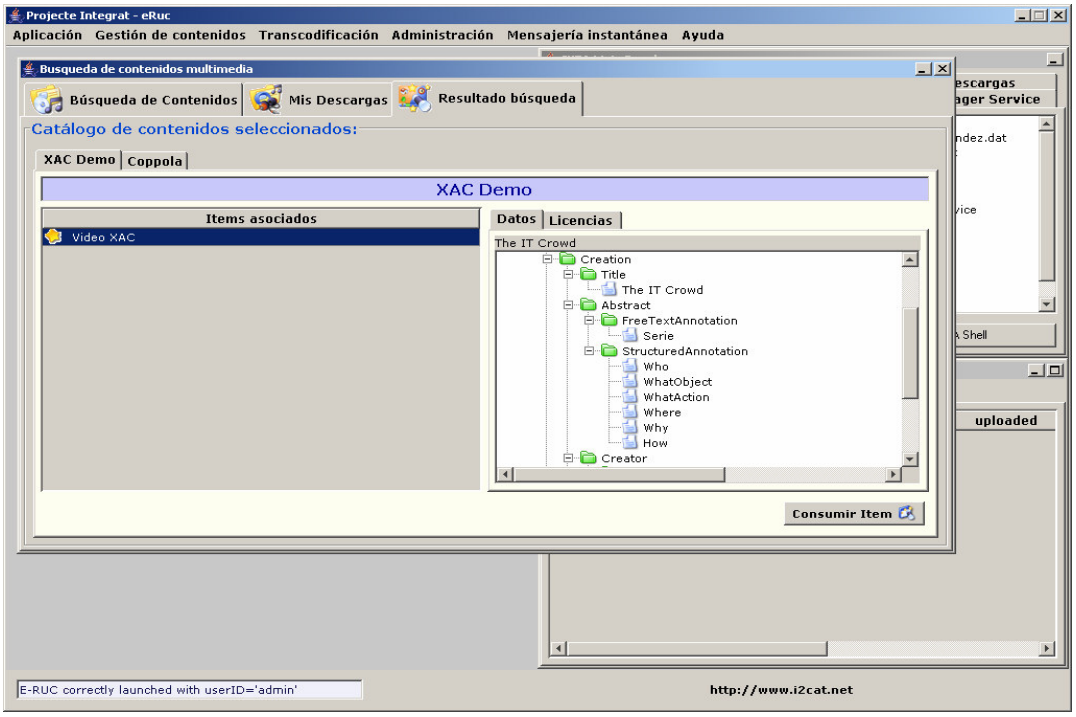


Il·lustració 29. Finestra XFORM per la cerca de continguts multimèdia

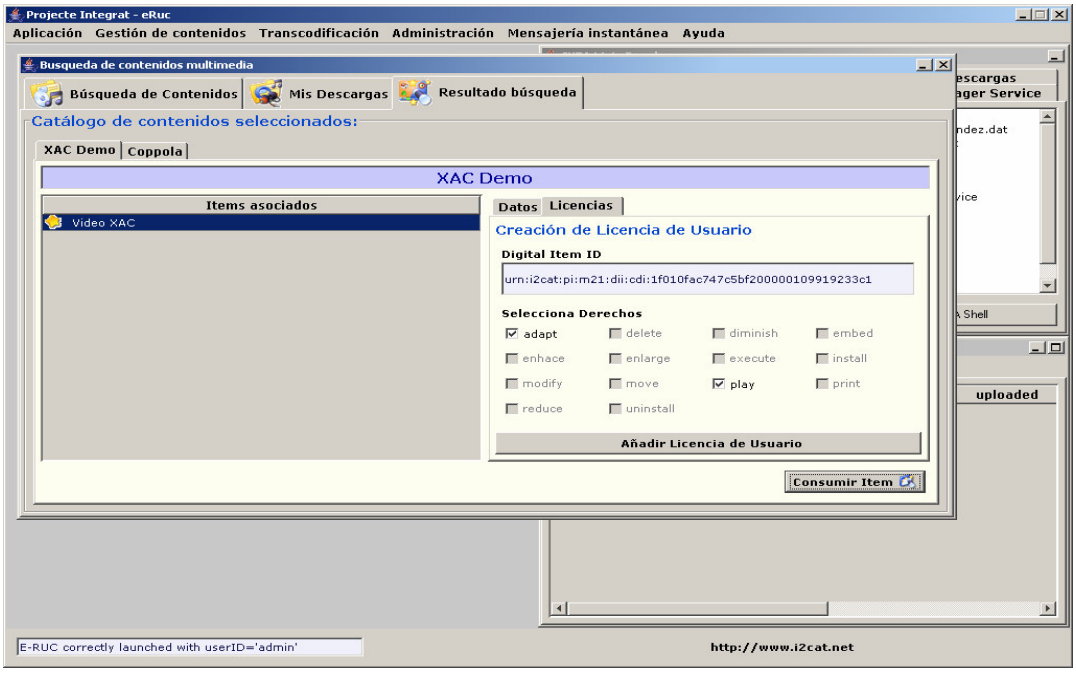
De forma anàloga a l'apartat 2.3.2.2 "Catalogació de continguts", podem fer una cerca cridant als tres mòduls principals de treball de l'e-Ruc, això és, el mòdul de metadades, el de llicències i el de descàrregues de contingut P2P.

La finestra pròpiament, es composa de dues solapes o pestanyes "Cerca de continguts" i "Les meves descàrregues" amb funcionalitat ben diferent.

Amb la primera, a partir de triar el tipus de recerca (si bàsica o avançada) que volem realitzar, ens dóna els resultats vinculats a la paraula clau o criteris de recerca emprats. Un cop seleccionem el DI que ens interessa obtenir de la xarxa, només cal fer clic sobre ell i se'ns obrirà una tercera solapa amb tota la informació relativa a les metadades que singularitzen tal fitxer [figura 11] i sobre els tipus de drets i les llicències vinculades al fitxer en particular [figura 12].



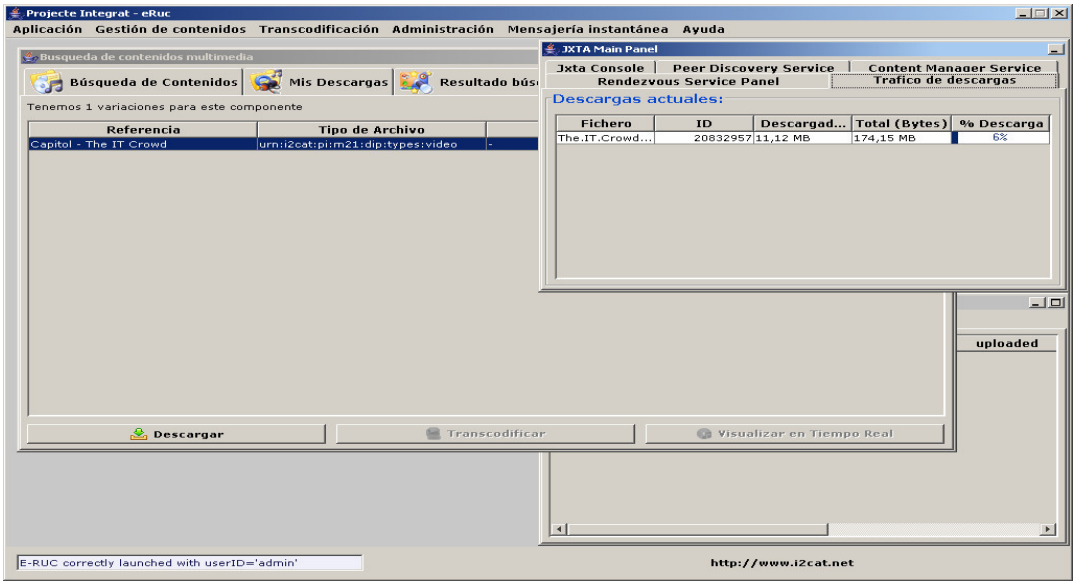
Il·lustració 30. Detall de les metadades associades a un fitxer qualsevol



Il·lustració 31. Detall del mòdul de llicències vinculades a un fitxer qualsevol de la xarxa

Un cop acceptades les condicions particulars sobre llicències i drets del DI que volem, només cal fer clic sobre el botó "consumir ítem". En aquell moment es realitza una crida al mòdul de descarregues P2P i de forma immediata es

produirà la descàrrega del DI de forma local en el format escollit dins de la carpeta descarregues de l'e-Ruc [figura 13].



Il·lustració 32. Detall del mòdul de descàrregues de continguts

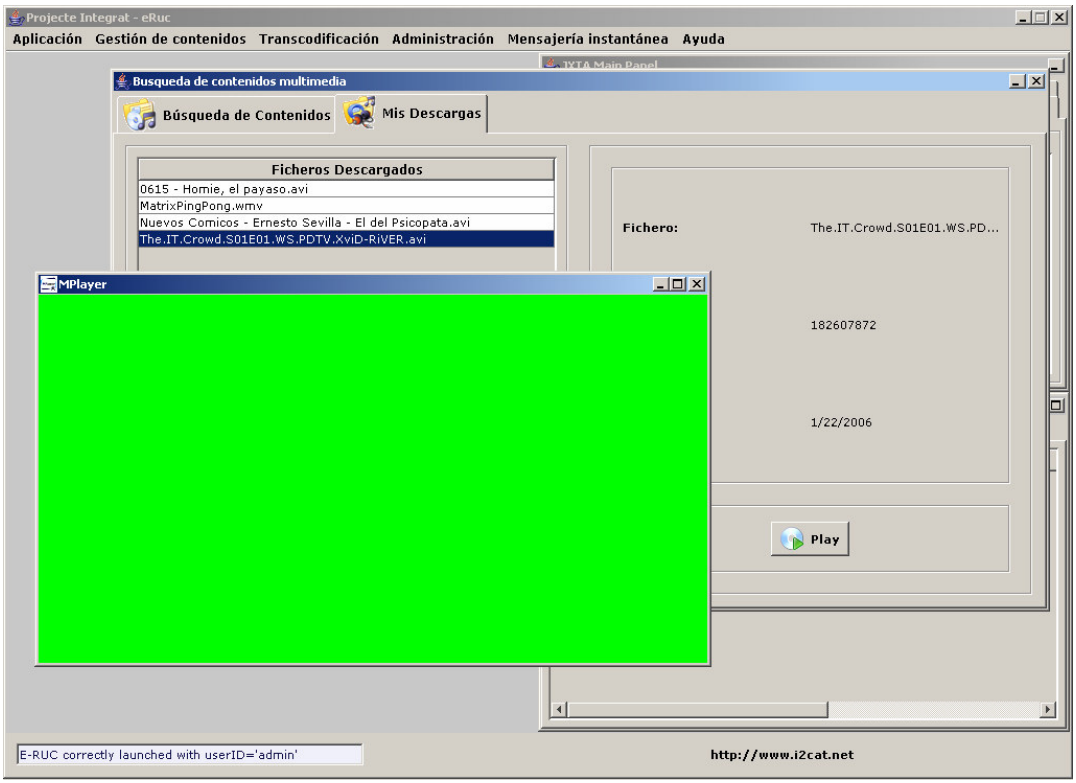
És important senyalar que en la versió definitiva de l'e-Ruc, es podrà realitzar, un cop tinguem els resultats d'una recerca, una previsualització en baixa qualitat del DI en format MPEG2 ó SIF(MPEG1), tot depenent de l'estat de la xarxa, per saber a priori si el vídeo que es vol adquirir compleix les nostres expectatives. En el cas de que finalment es vulgui adquirir i els drets i llicències estipulats ho permetin, es podrà descarregar el fitxer en tres formats diferents: un tancat (llest per l'emissió) en format MPEG-2 i un altre obert (per si es vol editar en posterioritat) en formats d'alta i baixa qualitat DV i M-JPEG.

8.1.2.5 CONTINGUTS DESCARREGATS DE LA XARXA

Per una altra banda, amb la segona solapa "Les meves descarregues" podem controlar els DI descarregats prèviament de la xarxa i que ja disposem de forma local, tot donant-nos informació relativa al fitxer: nom, data i tamany, a més de poder realitzar una visualització del mateix.

Aquesta visualització es realitza cridant a l'aplicació externa Movie Player (MPLAYER) que cal tenir prèviament instal·lada.

Fins que no surti la versió de l'e-Ruc amb el reproductor de medis integrat , caldrà en un primer moment indicar al programa la ruta per tal de trobar l'executable Mplayer.exe, en aquest moment apareixerà una finestra mostrant el DI escollit.

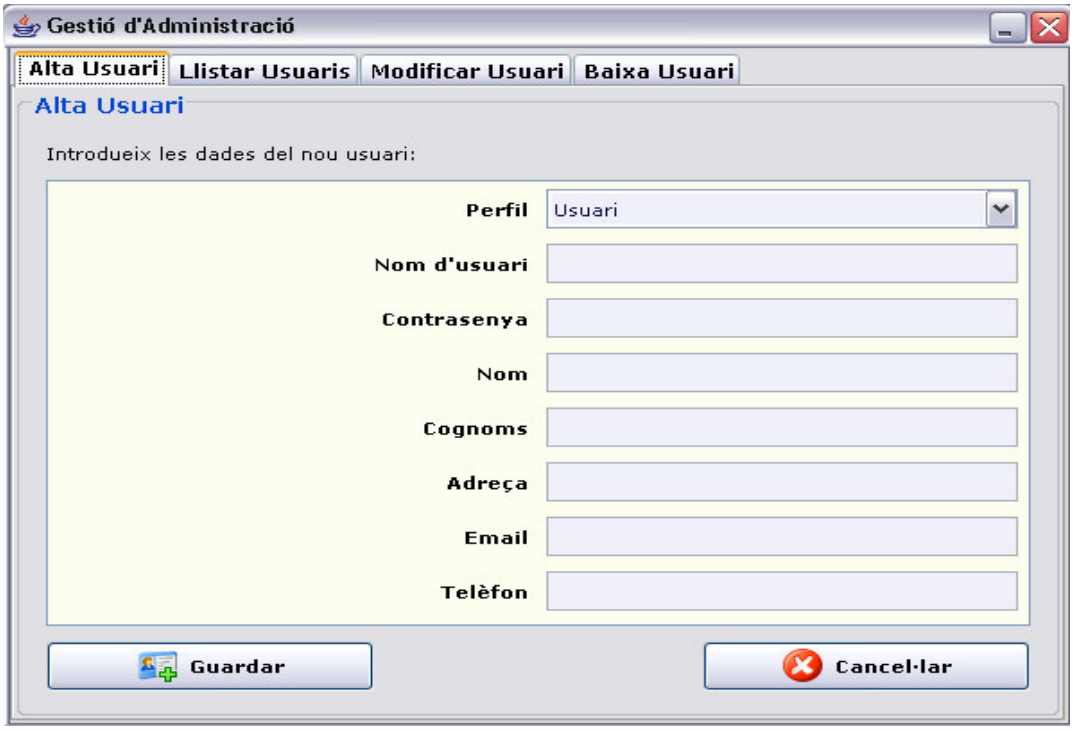


Il·lustració 33. Detall de la visualització d'un DI en local amb l'MPlayer

8.1.2.6 ADMINISTRACIÓ

En la versió definitiva de l'e-Ruc versió XAC, aquesta funció únicament estarà present per l'administrador global de la xarxa ja que serà l'encarregat de gestionar totes les altes i baixes de la base de dades dels usuaris així com establir els permisos dels que gaudiran cadascun d'ells.

Des de la barra de menús, es pot realitzar una gestió de la base de dades (BD) [figura 15] o un esborrat (reset de la BD) de la mateixa. La gestió d'aquesta base de dades, com s'ha comentat vindrà donada per l'administrador o equivalent que gaudeixi dels suficients privilegis com per donar d'alta, modificar o inclús esborrar usuaris.



Il·lustració 34. Detall de la gestió dels usuaris de la xarxa

8.2 Annex B: WSDL dels tres serveis WEB implementats

ApiTranscodeModule11.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn:ApiTranscodeModule11"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="urn:ApiTranscodeModule11"
xmlns:intf="urn:ApiTranscodeModule11"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.3
Built on Oct 05, 2005 (05:23:37 EDT)-->

   <wsdl:message name="getResourcesResponse">

      <wsdl:part name="getResourcesReturn" type="soapenc:string"/>

   </wsdl:message>

   <wsdl:message name="getListTXMediaServersResponse">

      <wsdl:part name="getListTXMediaServersReturn"
type="soapenc:string"/>

   </wsdl:message>

   <wsdl:message name="getResourcesRequest">

      <wsdl:part name="token" type="soapenc:string"/>

      <wsdl:part name="signature" type="soapenc:string"/>

   </wsdl:message>

   <wsdl:message name="controlRequest">

      <wsdl:part name="serverType" type="soapenc:string"/>

      <wsdl:part name="output" type="soapenc:string"/>

      <wsdl:part name="control" type="soapenc:string"/>

      <wsdl:part name="token" type="soapenc:string"/>

      <wsdl:part name="signature" type="soapenc:string"/>

   </wsdl:message>

   <wsdl:message name="getListTXMediaServersRequest">

      <wsdl:part name="token" type="soapenc:string"/>

      <wsdl:part name="signature" type="soapenc:string"/>

   </wsdl:message>
```

```
   <wsdl:message name="controlResponse">

      <wsdl:part name="controlReturn" type="soapenc:string"/>

   </wsdl:message>

   <wsdl:message name="createTXMediaServerResponse">

      <wsdl:part name="createTXMediaServerReturn"
type="soapenc:string"/>

   </wsdl:message>

   <wsdl:message name="createTXMediaServerRequest">

      <wsdl:part name="serverType" type="soapenc:string"/>

      <wsdl:part name="output" type="soapenc:string"/>

      <wsdl:part name="input" type="soapenc:string"/>

      <wsdl:part name="protoTx" type="soapenc:string"/>

      <wsdl:part name="token" type="soapenc:string"/>

      <wsdl:part name="signature" type="soapenc:string"/>

   </wsdl:message>

   <wsdl:portType name="ApiTranscodeModule11">

      <wsdl:operation name="createTXMediaServer"
parameterOrder="serverType output input protoTx token signature">

         <wsdl:input message="impl:createTXMediaServerRequest"
name="createTXMediaServerRequest"/>

         <wsdl:output message="impl:createTXMediaServerResponse"
name="createTXMediaServerResponse"/>

      </wsdl:operation>

      <wsdl:operation name="getListTXMediaServers"
parameterOrder="token signature">

         <wsdl:input message="impl:getListTXMediaServersRequest"
name="getListTXMediaServersRequest"/>

         <wsdl:output message="impl:getListTXMediaServersResponse"
name="getListTXMediaServersResponse"/>

      </wsdl:operation>

      <wsdl:operation name="getResources" parameterOrder="token
signature">

         <wsdl:input message="impl:getResourcesRequest"
name="getResourcesRequest"/>

         <wsdl:output message="impl:getResourcesResponse"
name="getResourcesResponse"/>
```

```
      </wsdl:operation>

      <wsdl:operation name="control" parameterOrder="serverType output
control token signature">

         <wsdl:input message="impl:controlRequest"
name="controlRequest"/>

         <wsdl:output message="impl:controlResponse"
name="controlResponse"/>

      </wsdl:operation>

   </wsdl:portType>

   <wsdl:binding name="ApiTranscodeModuleSoapBinding"
type="impl:ApiTranscodeModule11">

      <wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>

      <wsdl:operation name="createTXMediaServer">

         <wsdlsoap:operation soapAction=""/>

         <wsdl:input name="createTXMediaServerRequest">

            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ApiTranscodeModule11" use="encoded"/>

         </wsdl:input>

         <wsdl:output name="createTXMediaServerResponse">

            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ApiTranscodeModule11" use="encoded"/>

         </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="getListTXMediaServers">

         <wsdlsoap:operation soapAction=""/>

         <wsdl:input name="getListTXMediaServersRequest">

            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ApiTranscodeModule11" use="encoded"/>

         </wsdl:input>

         <wsdl:output name="getListTXMediaServersResponse">

            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ApiTranscodeModule11" use="encoded"/>
```

```
         </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="getResources">

         <wsdlsoap:operation soapAction=""/>

         <wsdl:input name="getResourcesRequest">

            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ApiTranscodeModule11" use="encoded"/>

         </wsdl:input>

         <wsdl:output name="getResourcesResponse">

            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ApiTranscodeModule11" use="encoded"/>

         </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="control">

         <wsdlsoap:operation soapAction=""/>

         <wsdl:input name="controlRequest">

            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ApiTranscodeModule11" use="encoded"/>

         </wsdl:input>

         <wsdl:output name="controlResponse">

            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:ApiTranscodeModule11" use="encoded"/>

         </wsdl:output>

      </wsdl:operation>

   </wsdl:binding>

   <wsdl:service name="ApiTranscodeModule11Service">

      <wsdl:port binding="impl:ApiTranscodeModuleSoapBinding"
name="ApiTranscodeModule">

         <wsdlsoap:address
location="http://localhost:8080/axis/services/ApiTranscodeModule"/>

      </wsdl:port>
```

```
   </wsdl:service>

</wsdl:definitions>
```

UploadService.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn:UploadService"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="urn:UploadService" xmlns:intf="urn:UploadService"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.3
Built on Oct 05, 2005 (05:23:37 EDT)-->

   <wsdl:message name="publishThumbnailResponse">

      <wsdl:part name="publishThumbnailReturn" type="soapenc:string"/>

   </wsdl:message>

   <wsdl:message name="storeDocumentXMLResponse">

      <wsdl:part name="storeDocumentXMLReturn" type="soapenc:string"/>

   </wsdl:message>

   <wsdl:message name="storeDocumentXMLRequest">

      <wsdl:part name="dh" type="apachesoap:Source"/>

      <wsdl:part name="filename" type="soapenc:string"/>

      <wsdl:part name="token" type="soapenc:string"/>

      <wsdl:part name="signature" type="soapenc:string"/>

   </wsdl:message>

   <wsdl:message name="publishThumbnailRequest">

      <wsdl:part name="dh" type="apachesoap:DataHandler"/>

      <wsdl:part name="filename" type="soapenc:string"/>

      <wsdl:part name="token" type="soapenc:string"/>

      <wsdl:part name="signature" type="soapenc:string"/>

   </wsdl:message>

   <wsdl:message name="publishResourceResponse">

      <wsdl:part name="publishResourceReturn" type="soapenc:string"/>
```

```
   </wsdl:message>

   <wsdl:message name="publishResourceRequest">

      <wsdl:part name="dh" type="apachesoap:DataHandler"/>

      <wsdl:part name="filename" type="soapenc:string"/>

      <wsdl:part name="token" type="soapenc:string"/>

      <wsdl:part name="signature" type="soapenc:string"/>

   </wsdl:message>

   <wsdl:portType name="UploadService">

      <wsdl:operation name="publishResource" parameterOrder="dh
filename token signature">

         <wsdl:input message="impl:publishResourceRequest"
name="publishResourceRequest"/>

         <wsdl:output message="impl:publishResourceResponse"
name="publishResourceResponse"/>

      </wsdl:operation>

      <wsdl:operation name="publishThumbnail" parameterOrder="dh
filename token signature">

         <wsdl:input message="impl:publishThumbnailRequest"
name="publishThumbnailRequest"/>

         <wsdl:output message="impl:publishThumbnailResponse"
name="publishThumbnailResponse"/>

      </wsdl:operation>

      <wsdl:operation name="storeDocumentXML" parameterOrder="dh
filename token signature">

         <wsdl:input message="impl:storeDocumentXMLRequest"
name="storeDocumentXMLRequest"/>

         <wsdl:output message="impl:storeDocumentXMLResponse"
name="storeDocumentXMLResponse"/>

      </wsdl:operation>

   </wsdl:portType>

   <wsdl:binding name="UploadServiceSoapBinding"
type="impl:UploadService">

      <wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>

      <wsdl:operation name="publishResource">

         <wsdlsoap:operation soapAction=""/>
```

```
         <wsdl:input name="publishResourceRequest">

            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:UploadService" use="encoded"/>

         </wsdl:input>

         <wsdl:output name="publishResourceResponse">

            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:UploadService" use="encoded"/>

         </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="publishThumbnail">

         <wsdlsoap:operation soapAction=""/>

         <wsdl:input name="publishThumbnailRequest">

            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:UploadService" use="encoded"/>

         </wsdl:input>

         <wsdl:output name="publishThumbnailResponse">

            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:UploadService" use="encoded"/>

         </wsdl:output>

      </wsdl:operation>

      <wsdl:operation name="storeDocumentXML">

         <wsdlsoap:operation soapAction=""/>

         <wsdl:input name="storeDocumentXMLRequest">

            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:UploadService" use="encoded"/>

         </wsdl:input>

         <wsdl:output name="storeDocumentXMLResponse">

            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:UploadService" use="encoded"/>

         </wsdl:output>

      </wsdl:operation>
```

```
   </wsdl:binding>

   <wsdl:service name="UploadServiceService">

      <wsdl:port binding="impl:UploadServiceSoapBinding"
name="UploadService">

         <wsdlsoap:address
location="http://localhost:8080/axis/services/UploadService"/>

      </wsdl:port>

   </wsdl:service>

</wsdl:definitions>
```

DownloadService.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn:DownloadService"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="urn:DownloadService" xmlns:intf="urn:DownloadService"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.3
Built on Oct 05, 2005 (05:23:37 EDT)-->

   <wsdl:message name="downloadResourceRequest">

      <wsdl:part name="reference" type="soapenc:string"/>

      <wsdl:part name="token" type="soapenc:string"/>

      <wsdl:part name="signature" type="soapenc:string"/>

   </wsdl:message>

   <wsdl:message name="downloadResourceResponse">

      <wsdl:part name="downloadResourceReturn" type="xsd:anyType"/>

   </wsdl:message>

   <wsdl:portType name="DownloadService">

      <wsdl:operation name="downloadResource"
parameterOrder="reference token signature">

         <wsdl:input message="impl:downloadResourceRequest"
name="downloadResourceRequest"/>

         <wsdl:output message="impl:downloadResourceResponse"
name="downloadResourceResponse"/>

      </wsdl:operation>

   </wsdl:portType>
```

```
   <wsdl:binding name="DownloadServiceSoapBinding"
type="impl:DownloadService">

      <wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>

      <wsdl:operation name="downloadResource">

         <wsdlsoap:operation soapAction=""/>

         <wsdl:input name="downloadResourceRequest">

            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:DownloadService" use="encoded"/>

         </wsdl:input>

         <wsdl:output name="downloadResourceResponse">

            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:DownloadService" use="encoded"/>

         </wsdl:output>

      </wsdl:operation>

   </wsdl:binding>

   <wsdl:service name="DownloadServiceService">

      <wsdl:port binding="impl:DownloadServiceSoapBinding"
name="DownloadService">

         <wsdlsoap:address
location="http://localhost:8080/axis/services/DownloadService"/>

      </wsdl:port>

   </wsdl:service>

</wsdl:definitions>
```

8.3 Annex C: Funcionament Api VideoLAN

Per configurar aquesta Api a l'equip de la XAC s'ha utilitzat la configuració de l'equip del PI. Aquesta aplicació està basada en 4 classes: *CapaControl, ListaSrvVideo, ServidorVlc* i *HiloEjecucion*. El funcionament d'aquestes quatre classes es el següent:

- CapaControl realitza la gestió principal de la interfície. S'encarrega d'organitzar les inicialitzacions i ordenar l'enviament de les dades de control als servidors (inicia els paràmetres principals com el port, o el numero màxim de servidors..., processa les dades per iniciar o localitzar un nou servidor, envia el resultat OK o Error, etc.).
- ListaSrvVideo fa la funció de contenidor dels servidors. Aquesta classe esta formada per un array de servidors i un conjunt de funcions per manipular-los: crea, borra, busca o arranca un servidor de la llista.
- ServidorVlc és la classe que interacciona amb vlc. Les seves funcions principals son la generació de la línea d'execució (incloent les funcions de verificació de paràmetres) i el control remot de vlc.
- HiloEjecucion hereta de la classe Thread i te la funció de llançar un nou procés cap al vlc al sistema. Cada objecte ServidorVlc inclou un objecte HiloEjecución.

Amb la interacció d'aquestes classes s'aconsegueix les funcions bàsiques que es necessiten en aquest projecte, com arrancar una interfície vlc, creació d'un flux o control/destrucció d'un flux.

El funcionament d'aquesta aplicació es basa en tres estats: espera de dades, processament de dades i enviament del resultat de processament (el servidor de media pot rebre OK o ERROR amb la descripció corresponent).

Procediment d'arrencada de la interfície:

1. Execució de la línea de comandes (Ex. Java CapaControl port)
2. Inicialització dels paràmetres de la interfície port i servidors màxims
3. Inicialització del socket (s'estableix LISTEB al port xxx de totes les IPs configurades a l'equip)

En aquest procediment només interfereix la classe Capa de control. Aquesta rep els arguments de la línea de comandes i entra al mètode que inicia la interfície (i aquest a invoca al mètode que queda a l'escolta fins rebre alguna cadena).

Procediment de creació d'un servidor:

1. El Media Server envia una cadena de inicialització
2. La interfície rep i verifica el número de paràmetres
3. Processa la cadena
    a. Traspàs de la cadena a la classe ListaSrvVideo
    b. La classe ListaSrvVideo crea el servidor
4. Enviament del resultat de l'operació en forma de cadena de caràcters

Al rebre les dades, aquestes es retornen al mètode que inicialitza la interfície que les passa al mètode que processa aquestes dades. Aquest, crea el servidor a la llista de servidors i l'arranca. En aquestes dues accions s'accedeix als mètodes de la classe ServidorVlc.

Procediment de control d'un servidor:

1. El media Server envia una cadena de control
2. La interfície la rep i verifica el número de paràmetres
3. Processa la cadena
    a. Localitza el servidor
    b. Traspàs de la cadena de control
4. Enviament del resultat de l'operació en forma de cadena de caràcters

En aquest cas segueix el mateix procediment anterior, amb la diferencia de que l'accés es directe al servidor des del mètode de processament de la comanda. Aquest mètode utilitza les funcions de la classe ListaSrvVideo per localitzar el servidor i a continuació accedeix a la funció *control* de l'objecte ServidorVlc localitzat.

Procediment de destrucció d'un servidor:

1. El Media Server envia una cadena de control
2. La interfície la rep i verifica el número de paràmetres
3. Processa la cadena
    a. Localització del servidor
    b. Detecció de la directriu *kill*
    c. Eliminació del servidor de la llista i destrucció del fil d'execució
4. Enviament del resultat de l'operació en forma de cadena de caràcters

La detecció de la directriu *kill* es realitza a la funció de processar la comanda, que una vegada localitzat el servidor, accedeix a la funció que s'encarrega d'esborrar el servidor i que es troba a la classe ListaSrvVideo. Aquesta funció esborra i reinicialitza la posició que ocupa el servidor.

8.4 Annex D: Configuració del Nagios per la gestió de l'equip

L'equip utilitzat per configurar la gestió ha estat l'equip del MediaCAT que ja te instal·lats els scripts de Nagios, per tant, només ha estat necessari configurar els arxius corresponents per afegir tots quatre serveis a monitoritzar i crear els scripts per reiniciar cada servei.

L'equip que conté la gestió de Nagios te una IP privada de MediaCAT, peró també està accessible per un port concret amb la IP 84.88.32.42 (accés amb password). Els arxius de configuració de Nagios es troben a la carpeta:

/usr/nagios/etc

Dintre de la carpeta Nagios, es troben també altres subcarpetes que contenen els scripts que executaran els arxius de configuració de Nagios.

Els arxius de configuració que s'han de modificar són:

- Hosts.cfg → conté tots els equips associant nom amb IP.
- Checkcommands.cfg → conté les comandes que executarà Nagios per fer el chequeig d'un servei
- Services.cfg → conté tots els serveis que es monitoritzen al Nagios

S'edita primer l'arxiu amb els equips i afegim l'equip de la XAC:

```
####################### EQUIPO XAC ############################

define host{
      host_name                   XAC  # nom de l'equip
      alias                       XAC # alias associat a l'equip
      address                     84.88.32.44 # IP de l'equip
      parents                     SMC.11 # Equip de xarxa del que depèn
      check_command               check-host-alive # Comanda que s'executarà
      checks_enabled              1 # 0/1 en funció si es permeten chequejos o no
      max_check_attempts          5 # Numero de vegades que es chequeja un servei
                  abans d'enviar una notificació i en cas de que l'estat no sigui OK
      notification_interval       120 # temps d'espera en segons per tornar a notificar
                  un estat de down
      notification_period         24x7 #hores al dia/dies a la setmana que es pot notifica
      notification_options        d # notificar només si l'estat es down
}
```

Una vegada afegit el host, configurem els quatre serveis que es volen monitoritzar:

```
############ SERVICIOS EQUIPO XAC ##################


define service{
      host_name                   XAC
      service_description         Servei SSH
      check_command               check_ssh  # Chequejem que el daemon SSH estigui
funcionant
      max_check_attempts          3
```

```
        normal_check_interval       5
        retry_check_interval        5
        active_checks_enabled       1
        check_period                24x7
        notification_interval       30
        notification_period         24x7
        notification_options        c
        contact_groups              silvia
}

define service{
        host_name               XAC
        service_description     Ping
        check_command           check-host-alive # Comanda per fer pings a l'equip
        max_check_attempts      3
        normal_check_interval   5
        retry_check_interval    5
        active_checks_enabled   1
        check_period            24x7
        notification_interval   30
        notification_period     24x7
        notification_options    c
        contact_groups          silvia
}

define service{
        host_name               XAC
        service_description     Puerto Tomcat
        check_command           check_tcp!8080 # Es comprova que el port 8080 estigui
actiu
        max_check_attempts      3
        normal_check_interval   5
        retry_check_interval    5
        active_checks_enabled   1
        check_period            5a6am
        notification_interval   30
        notification_period     24x7
        notification_options    c
        contact_groups          silvia
}

define service{
        host_name               XAC
        service_description     Puerto VideoLAN
        check_command           check_tcp!9912 # Es comprova que el port 9912 de
VideoLAN, estigui actiu
        max_check_attempts      3
        normal_check_interval   5
        retry_check_interval    5
        active_checks_enabled   1
        check_period            5a6am
        notification_interval   30
        notification_period     24x7
        notification_options    c
        contact_groups          silvia
}
```

Ara ja es tenen tots els serveis associats a l'equip XAC. Es comprova que els arxius, per fer els *checks,* existeixin. Aquests arxius es troben a la carpeta:

/usr/nagios/libexec

Per aquest equip es necessiten tres checks: per fer un ping, per comprovar el ssh i per comprovar els dos ports TCP:

```
        ################  Check PING

        define command{
              command_name    check-host-alive
              command_line          /usr/nagios/libexec/check_ping   -H   $HOSTADDRESS$   -w
100,25% -c 1000,30%
        }

        ################  Check SSH

        define command{
              command_name    check_ssh
              command_line    /usr/nagios/libexec/check_ssh -t 7 -p 22 $HOSTADDRESS$
        }

        ################  Check SSH

        define command{
              command_name    check_tcp
              command_line     /usr/nagios/libexec/check_tcp -H $HOSTADDRESS$ -p $ARG1$
-w 5 -c 10
        }
```

Una vegada tot configurat correctament, reiniciem el daemon de Nagios i ja trobem els serveis a la pàgina principal de Nagios:

/etc/init.d/nagios restart

Un altre funcionalitat molt interessant que ens proporciona Nagios és la de executar un arxiu o comanda en cas que hi hagi un avis de un problema en un servei, és a dir, que l'estat del servei sigui *warning* o *critical.* S'aprofitarà llavors aquesta funcionalitat per reiniciar els serveis que es vegin afectats en un determinat moment. Per fer-ho, Nagios, te associada una carpeta anomenada *eventhandlers* on es guarden els scripts que s'han d'executar en cas d'algun problema. S'utilitza llavors l'arxiu *checkcommands.cfg* per cridar aquests scripts.

Per utilitzar aquesta aplicació es necessari disposar d'un aplicació com el *nrpe* (veure [8]) que vagi corrent a l'equip remot per a que sigui el propi equip que te el problema el que reinicïi els serveis, i no l'equip de Nagios. Per tant, s'instal·la aquesta aplicació i es configura per a que accepti les peticions de l'equip de Nagios.

Es crea llavors un arxiu per reiniciar el servei del Tomcat, que s'anomenarà *reboot_tomcat* i, en aquest cas, com que no es un script que executarà el Nagios, sinó que serà el propi equip de la XAC, es col·locarà a un directori de l'equip de la XAC que després s'especificarà a l'arxiu de configuració de NRPE, nrpe.cfg que està a la XAC.

```
command[reboot_tomcat]=/etc/nagios-plugins/config/reboot_tomcat
```

L'script que s'ha creat està fet en *bash,* per tant és tan senzill com:

```
#!/bin/bash

/usr/local/tomcat5/bin/catalina.sh stop
/usr/local/tomcat5/bin/catalina.sh start
```

Pel que fa al Nagios, s'han d'afegir dues linies a la configuració del servei de la XAC que monitoritza el Tomcat per especificar que es vol fer en cas d'error del servei. Per tant queda així:

```
define service{
        host_name                       XAC
        service_description             Puerto Tomcat
        check_command                   check_tcp!8080
        event_handler                   check_nrpe
        event_handler_enabled           1
        max_check_attempts              3
        normal_check_interval           5
        retry_check_interval            5
        active_checks_enabled           1
        check_period                    5a6am
        notification_interval           30
        notification_period             24x7
        notification_options            c
        contact_groups                  silvia
}
```

I finalment afegir la comanda del NRPE a l'arxiu *checkcommands.cfg*:

```
define command{
        command_name    check_nrpe
        command_line    /usr/nagios/libexec/check_nrpe -H $HOSTADDRESS$ -p 6666 -t
30 -c reboot_tomcat
}
```

L'opció – *c* indica la comanda que després el daemon de NRPE buscarà a l'arxiu de configuració en local, en la XAC, i executarà.